(72) Inventors:
      • **Nguyen, Tai H.**
        Houston, Texas 77082 (US)
      • **Jones, Jason A.T.**
        Houston, Texas 77042 (US)
      • **Bradley, Jonathan G.**
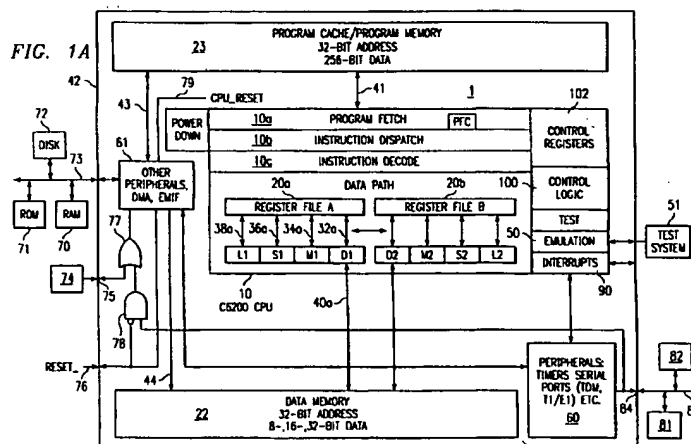   **Houston, Texas 77083 (US)**

      • **Seshan, Natarajan**
        Houston Texas 77063-1234 (US)
      • **Quay, Jeffrey R.**
        Royse City Texas 75189 (US)
      • **Williams, Kenneth L.**
        Sherman Texas 75090 (US)
      • **Moody, Michael J.**
        McKinney, Texas 75070 (US)
      • **Simar, Laurence R.Jr.**
        Richmond, Texas 77469 (US)
      • **Scales, Richard H.**
        06271 Villeneuve-Loubet (FR)

(74) Representative: **Potter, Julian Mark et al**
      D. Young & Co.,
      21 New Fetter Lane
      London EC4A 1DA (GB)

(54) **A digital signal processor with peripheral devices and external interfaces**

(57) A Digital Signal Processor is described which
has a variety of on-chip peripheral devices and a variety
of external interfaces. The peripheral devices include a
multi-channel serial port, a multi-channel direct memory
interface, and a timer. The external interfaces include a
host port interface and an extended memory interface.

FIG. 1A

Description

Technical Field of the Invention

5   **[0001]**   This invention relates to microprocessors, and particularly relates to microprocessor integrated circuits which include an on-chip peripheral devices and various external interface ports.

Background of the Invention

10   **[0002]**   Microprocessor designers have increasingly endeavored to improve performance in various microprocessors by increasing clock speeds and adding parallelism. Large blocks of random access memory (RAM) are included within the microprocessor for data storage and for program storage in order to reduce memory access times. Various types of input/output devices are included within a microprocessor integrated circuit in order to reduce total system chip count and cost. Serial ports are an example of a class of input/output devices that are commonly included with microproc-
15   essors.
     [0003]   An object of the present invention is to provide improvements in the operation and control of on-chip peripheral devices and external interfaces for microprocessor integrated circuits.

Summary of the Invention

20
     [0004]   In general, one form of the invention is a data processing device including a central processing unit (CPU) for executing instructions, a memory circuit connected to the central processing unit for storing the instructions which are executed by the CPU, and a serial port interface circuit connected to the CPU, operable to transmit and receive data with dual phase frames, wherein each phase has a different set of parameters.

25   **[0005]**   Another form of the invention is a data processing device including a central processing unit (CPU) for executing instructions, a memory controller connected to said central processing unit, and a memory circuit connected to said memory controller. The data processing device may further include an external memory interface circuit connected to said memory controller. The data processing device may further include a host port interface connected to said direct memory access controller. The data processing device may further include a peripheral bus controller connected to
30   said memory controller, and an external memory interface controller connected to said peripheral bus controller. The data processing device may further include at least one external signal for defining the type of memory located at a reset address for said central processing unit. The data processing device may further include at least one external signal for identifying the size and location of said memory circuit. The data processing device may further include a programmable timer connected to said peripheral bus controller. The data processing device may further include a
35   programmable phase-locked loop circuit for providing clock signals to said central processing unit. The data processing device may further include a power down circuit for programmably preventing clocking signals from reaching preselected portions of said device.
     [0006]   Another form of the invention is a data processing device including a central processing unit operable to execute software instructions stored in a program memory circuit connected to the central processing unit, a memory
40   circuit operable to store data to be processed by the processing device, a direct memory access (DMA) controller having read address circuitry and write address circuitry, operable to transfer data from or to the memory circuit, the DMA controller further having DMA interrupt circuitry operable to interrupt the central processing unit. The device may further include a peripheral device having address generation circuitry operable transfer data to or from the memory circuit. The device may further include auxiliary channel control circuitry operable to cause the peripheral device to
45   transfer a first data word to the memory circuit using the address generation circuitry of the peripheral device and to interrupt the central processor using the DMA interrupt circuitry of the DMA controller.
     [0007]   Other device, system and method forms of the invention are also disclosed and claimed herein. Other objects of the invention are disclosed and still other objects will be apparent from the disclosure herein.

50   Brief Description of the Drawings

     [0008]   Other features and advantages of the present invention will become apparent by reference to the following detailed description when considered in conjunction with the accompanying drawings, in which:

55   Figure 1A is a block diagram of a microprocessor which includes an embodiment of the present invention;
         Figure 1B and 1C are a more detailed block diagram of the microprocessor of Fig. 1A;
         Figure 2 is a block diagram of the execution units and register files of the microprocessor of Fig. 1A;
         Figures 3A and 3B illustrate two memory maps used by the microprocessor of Fig. 1A;

Figure 4 is a simplified block diagram of the microprocessor of Fig. 1A;

Figure 5 shows how the cache uses the fetch packet address from the CPU;

Figure 6 shows the directions of data flow as well as the master (requester) and slave (resource) relationships between the DMA and other microprocessor modules;

Figure 7 illustrates a DMA Global Data Register;

Figure 8 illustrates the DMA Channel Primary Control Register;

Figure 9 illustrates the DMA Channel Secondary Control Register;

Figure 10 illustrates a DMA Channel Transfer Counter;

Figure 11 shows a DMA Global Data Register;

Figure 12 illustrates the DMA Channel Source Address Register;

Figure 13 illustrates the Destination Address Register;

Figure 14 illustrates the DMA Global Data Register;

Figure 15 illustrates a DMA Global Data Register used for split addresses;

Figure 16 illustrates the DMA Global Control Register which specifies Priority Between Channels;

Figure 17 illustrates the logic of all enabled conditions that form the DMA_INTx signal;

Figure 18 shows the internal data movement paths of the DMA controller including data buses and internal holding registers;

Figure 19 shows a simplified block diagram of host interface to the HPI;

Figure 19A illustrates an HPIC Register Diagram;

Figure 20 shows the equivalent circuit of the HCS-, HDS1 -, and H052-inputs;

Figure 21 shows HPI access timing for the cases when HAS- is not used;

Figure 22 shows HPI access timing for the cases when HAS- is used;

Figure 23 shows a simplified block diagram of the EMIF with some of its I/O signals and signal paths;

Figure 24 shows the EMIF Global Control Register;

Figure 25 shows the (four) CE Space Control Registers spaces supported by the EMIF;

Figure 26 illustrates the EMIF SDRAM Control Register;

Figure 27 shows the SDRAM refresh period register;

Figure 28 shows 16M bit SDRAM interfaces;

Figure 29 shows 64 Mbit SDRAM interfaces;

Figure 30 shows SDRAM refresh timings;

Figure 31 depicts the SDRAM mode register set;

Figure 32 depicts DRAM deactivation;

Figure 33 depicts an SDRAM with CAS latency 3, burst length 1, read;

Figure 34 depicts an SDRAM burst length 1, write;

Figure 35 shows the EMIF interfaces directly to industry standard synchronous burst SRAMs (SBSRAMs);

Figure 36 shows a 16 word read of an SBSRAM;

Figure 37 shows a 6 word write of an SBSRAM;

Figure 38 shows an EMIF interface to standard SRAM;

Figure 39 shows an EMIF interface to FIFOs;

Figure 40 shows EMIF interfaces to 8-bit ROM;

Figure 41 shows EMIF interfaces to 16-bit ROM;

Figure 42 shows EMIF interfaces to 32-bit ROM;

Figure 43 illustrates three asynchronous timings;

Figure 44 is a block diagram of multi-channel serial port (MCSP);

Figure 45 shows the Serial Port Control Register (SPCR);

Figure 46 shows the Pin Control Register;

Figure 47 shows the Receive Control Register;

Figure 48 shows the Transmit Control Register;

Figure 49 shows typical operation of the MCSP clock and frame sync signals;

Figure 50 shows how data clocked by an external serial device using a rising edge may be sampled by the MCSP receiver with the falling edge of the same clock;

Figure 51 shows an example of a frame where the first phase consists of 2 words of 12 bits each followed by a second phase of 3 words of 8 bits each;

Figure 52 shows a situation where four 8-bit words are transferred in a single phase frame;

Figure 53 illustrates a data stream of a single phase frame consisting of one 32-bit data word;

Figure 54 shows the range of programmable data delay is zero to two bit-clocks ((R/X)DATDLY=00b -10b);

Figure 55 shows how the serial port essentially discards the framing bit from the data stream;

Figure 56 shows an example of the Audio Codec '97 (AC97) standard;

Detailed Description of Embodiments of the Invention

[0009]   Figure 1A is a block diagram of a microprocessor 1 which includes an embodiment of the present invention. Microprocessor 1 is a VLIW digital signal processor ("DSP"). In the interest of clarity, Figure 1A only shows those

portions of microprocessor 1 that are relevant to an understanding of an embodiment of the present invention. Details of general construction for DSPs are well known, and may be found readily elsewhere. For example, U.S. Patent 5,072,418 issued to Frederick Boutaud, et al, describes a DSP in detail and is incorporated herein by reference. U.S. Patent 5,329,471 issued to Gary Swoboda, et al, describes in detail how to test and emulate a DSP and is incorporated

5   herein by reference. Details of portions of microprocessor 1 relevant to an embodiment of the present invention are explained in sufficient detail hereinbelow, so as to enable one of ordinary skill in the microprocessor art to make and use the invention.

[0010]   In microprocessor 1 there are shown a central processing unit (CPU) 10, data memory 22, program memory 23, peripherals 60 and an external memory interface (EMIF) with a direct memory access (DMA) 61. CPU 10 further

10   has an instruction fetch/decode unit 10a-c , a plurality of execution units, including an arithmetic and load/store unit D1, a multiplier M1, an ALU/shifter unit S1, an arithmetic logic unit ("ALU") L1, a shared multiport register file 20a from which data are read and to which data are written. Decoded instructions are provided from the instruction fetch/decode unit 10a-c to the functional units D1, M1, S1, and L1 over various sets of control lines which are not shown. Data are provided to/from the register file 20a from/to to load/store units D1 over a first set of busses 32a, to multiplier M1 over

15   a second set of busses 34a, to ALU/shifter unit S1 over a third set of busses 36a and to ALU L1 over a fourth set of busses 38a. Data are provided to/from the memory 22 from/to the load/store units D1 via a fifth set of busses 40a. Note that the entire data path described above is duplicated with register file 20b and execution units D2, M2, S2, and L2. Instructions are fetched by fetch unit 10a from instruction memory 23 over a set of busses 41. Emulation unit 50 provides access to the internal operation of integrated circuit 1 which can be controlled by an external test system 51.

20   [0011]   Note that the memory 22 and memory 23 are shown in Figure 1 to be a part of a microprocessor 1 integrated circuit, the extent of which is represented by the box 42. The memories 22-23 could just as well be external to the microprocessor 1 integrated circuit 42, or part of it could reside on the integrated circuit 42 and part of it be external to the integrated circuit 42. Also, an alternate number of execution units can be used.

[0012]   When microprocessor 1 is incorporated in a data processing system, additional memory or peripherals may

25   be connected to microprocessor 1, as illustrated in Figure 1. For example, Random Access Memory (RAM) 70, a Read Only Memory (ROM) 71 and a Disk 72 are shown connected via an external bus 73. Bus 73 is connected to an External Memory Interface (EMIF) which is part of functional block 61 within microprocessor 42. A Direct Memory Access (DMA) controller is also included within block 61. The DMA controller is generally used to move data between memory and peripherals within microprocessor 1 and memory and peripherals which are external to microprocessor 1. Data can

30   be transferred from block 61 to program memory 23 via bus 43; data can be transferred to/from data memory 22 via bus 44. Other types of peripherals, such as timer 82, are connected via host port bus 80. A bus interface is contained within block 60 for host port bus 80.

[0013]   Several example systems which can benefit from aspects of the present invention are described in U.S. Patent 5,072,418, which was incorporated by reference herein, particularly with reference to Figures 2-18 of U.S. Patent

35   5,072,418. A microprocessor incorporating an aspect of the present invention to improve performance or reduce cost can be used to further improve the systems described in U.S. Patent 5,072,418. Such systems include, but are not limited to, industrial process controls, automotive vehicle systems, motor controls, robotic control systems, satellite telecommunication systems, echo canceling systems, modems, video imaging systems, speech recognition systems, vocoder-modem systems with encryption, and such.

40   [0014]   A description of various architectural features of the microprocessor of Fig. 1A is provided in co-assigned European Patent application Number 981011291.7. A description of a complete set of instructions for the microprocessor of Fig. 1A is also provided in this co-assigned application.

[0015]   Figures 1B and 1C are a more detailed block diagram of the microprocessor of Fig. 1A. Multi-channel Serial Port 120 is connected to peripheral bus 110.

45   [0016]   Figure 2 is a block diagram of the execution units and register files of the microprocessor of Fig. 1A and shows a more detailed view of the buses connecting the various functional blocks. In this figure, all data busses are 32 bits wide, unless otherwise noted. Bus 40a has an address bus DA1 which is driven by mux 200a. This allows an address generated by either load/store unit D1 12a or D2 12b to provide an address for loads or stores for register file 20a. Data Bus LD1 loads data from an address in memory 22 specified by address bus DA1 to a register in load unit D1

50   12a. Unit D1 12a may manipulate the data provided prior to storing it in register file 20a. Likewise, data bus ST1 stores data from register file 20a to memory 22. Load/store unit D1 12a performs the following operations: 32-bit add, subtract, linear and circular address calculations. Load/store unit D2 12b operates similarly to unit D1 12a, with the assistance of mux 200b for selecting an address.

[0017]   ALU unit L1 18a performs the following types of operations: 32/40 bit arithmetic and compare operations; left

55   most 1, 0, bit counting for 32 bits; normalization count for 32 and 40 bits; and logical operations. ALU L1 18a has input src1 for a 32 bit source operand and input src2 for a second 32 bit source operand. Input msb_src is an 8 bit value used to form 40 bit source operands. ALU L1 18a has an output dst for a 32 bit destination operands. Output msb_dst is an 8 bit value used to form 40 bit destination operands. Two 32 bit registers in register file 20a are concatenated to hold

a 40 bit operand. Mux 211 is connected to input srcl and allows a 32 bit operand to be obtained from register file 20a via bus 38a or from register file 20b via bus 210. Mux 212 is connected to input src2 and allows a 32 bit operand to be obtained from register file 20a via bus 38a or from register file 20b via bus 210. ALU unit L2 18b operates similarly to unit L1 18a.

[0018] ALU/shifter unit S1 16a performs the following types of operations: 32 bit arithmetic operations; 32/40 bit shifts and 32 bit bit-field operations; 32 bit logical operations; branching; and constant generation. ALU S1 16a has input srcl for a 32 bit source operand and input src2 for a second 32 bit source operand. Input msb_src is an 8 bit value used to form 40 bit source operands. ALU S1 16a has an output dst for a 32 bit destination operands. Output msb_dst is an 8 bit value used to form 40 bit destination operands. Mux 213 is connected to input src2 and allows a 32 bit operand to be obtained from register file 20a via bus 36a or from register file 20b via bus 210. ALU unit S2 16b operates similarly to unit S1 16a, but can additionally perform register transfers to/from the control register file 102.

[0019] Multiplier M1 14a performs 16 x 16 multiplies. Multiplier M1 14a has input srcl for a 32 bit source operand and input src2 for a 32 bit source operand. Multiplier M1 14a dst for a 32 bit destination operands. Mux 214 is connected to input src2 and allows a 32 bit operand to be obtained from register file 20a via bus 34a or from register file 20b via bus 210. Multiplier M2 14b operates similarly to multiplier M1 14a.

[0020] Figure 3A and 3B shows two memory maps for the microprocessor of Fig. 1A. The memory is byte addressable and its total address range is 4G bytes (corresponding to a 32-bit internal address). The memory map is divided between the internal program memory 23, internal data memory 22 and three external memory spaces and internal peripheral space. A section of internal memory can be accessed by the host port interface (HPI) 60.

[0021] The internal memory consists of 512k bits of on-chip program/cache memory 23 and 512k bits of on-chip data memory 22. The program memory, configurable as cache or program, is organized in 2k of 256-bit instruction fetch packets. The CPU 10 fetches all instructions one fetch packet at a time. The packets are processed at the maximum rate of eight 32-bit instructions per CPU cycle or at a minimum of one instruction per cycle. The internal data memory is byte addressable by the CPU (for reads as well as writes) and supports byte, half-word and full word transfers.

[0022] All external data accesses by CPU 10 or DMA 100 pass through the external memory interface (EMIF) 103 (see Figure lc). The external memory is divided into three spaces - CE0, CE1 and CE2. Each has a dedicated chip enable signal that is asserted during data access to or from the corresponding space. Each external space has assigned a separate internal peripheral bus register that determines the shape of the read/write cycle when accessing asynchronous memory.

[0023] In addition to asynchronous memory, CE0 and CE2 spaces can also interface to other types of memory. SBSRAM or SDRAM memory can be assigned to those two spaces by controlling signal levels on signal groups CE0_TYPE and CE2_TYPE (pins DC2-DC5) during Reset 76.

[0024] External memory space CE1 can only interface to asynchronous memory. However, while spaces CE0 and CE2 are always 32-bit wide, the CE1 memory space can also be configured to the width of 8 or 16 bits by controlling signal levels on signal group CE1_WIDTH pins. The EMIF automatically packs bytes and half-words into words during read cycles - a feature typically used when booting from an 8- or 16-bit EPROM. The CE1 memory space can be used for ROM interfacing because ROM cycles are similar to asynchronous SRAM read cycles. Note, that while the CE1 space is the only external memory space that allows read cycles from 8- or 16-bit wide memory, read cycles from any external memory space can access byte or half-word sized data from 32-bit wide external memory. The EMIF data write cycles can transfer bytes, half-words or words to external memory as well, using BE_ control signals for byte selects. Data read cycles always latch all 4 bytes (all 4 BE_'s active) and the CPU then extracts the appropriate bytes internally if the data size is less then 32 bits. Note, that EMIF writes requested by the program memory controller 104 or the DMA 100/101, are always 32-bits wide, as opposed to 8-, 16-, or 32-bit transfers originated by the data memory controller 105.

CACHE

Overview

[0025] As shown in Figure 4 , the Program Memory Controller:

Performs CPU and DMA requests to internal program memory and necessary arbitration.
Performs CPU requests to external memory through the external memory interface (EMIF).
Manages the internal program memory when configured as cache.

Internal Program Memory

[0026] The internal program memory includes 64K bytes of RAM. This memory can include 2K 256-bit fetch packets

which is equivalent to 16K 32-bit instructions. The CPU through the program memory controller has a single-cycle throughput 256-bit wide connection to internal program memory.

TABLE 1.

Internal Program Memory Modes		
Program Mode	AAPCC Value	Description
Mapped	000	Memory mapped. Cache Disabled. (Reset default.)
Cache Enable	010	Cache accessed and updated on reads.
Cache Freeze	011	Cache accessed but not updated on reads.
Cache Bypass	100	Cache not accessed or updated on reads. AAAAA
Reserved	Other	

[0027]   The internal program memory can be utilized in four modes selected by the Program Cache Control (PCC) field (bits 7--5) in the CPU Control and Status Register as shown in Table 1. The modes are:

1) Mapped: Depending on the memory map selected, the program memory either is located at either address:
    00000000-0000FFFF in Map 1.
    01400000-0140FFFF in Map 0.
    Refer to the Boot Configuration, Reset, and Memory Map on how to select the memory map. In this mode, program fetches to the internal program memory address range the fetch packet at that address. In any cache mode, CPU accesses to this address range return a fetch packet containing all NOPs. Mapped mode is the default state of the internal program memory at reset.

2) Enabled: In cache enabled mode, any initial program fetch of an address causes cache miss. In a cache miss, the fetch packet is loaded from external memory interface (EMIF) and stored in the internal cache memory one 32-bit instruction at a time. When all 8 instructions in the fetch packet are received, it is sent to the CPU for execution. During this period the CPU is halted in it entirety. The number of wait-states incurred depends on the type of external memory used, the state of that memory, and any contention for the EMIF with other requesters such as the DMA or CPU data access. Any subsequent read from a cached address will cause a cache hit and that fetch packet is sent to the CPU without any wait-states. On the change from program memory mode to cache enabled mode, the program cache is flushed. This mode transition is the only means to flush the cache.

3) Freeze: During a cache freeze, the cache retains its current state. A program read to a frozen cache is identical to a read to an enabled cache with the exception that on a cache miss the data read from the external memory interface is not stored in the cache. A subsequent read of the same address will also cause a cache miss and the data will again be fetched from external memory. Cache freeze can ensure that critical program data is not over-written in the cache.

4) Bypass: When the cache is bypassed, any program read will fetch data from external memory. The data is not stored in the cache memory. Like cache freeze, in cache bypass the cache retains its state. This mode can ensure that external program data is being fetched.

Cache Architecture

[0028]   The architecture of the cache is direct mapped. The 64K byte cache contains 2K fetch packets and thus 2K frames. (Often the terms block or line are used interchangeably with frame in cache terminology.) The width of the cache or frame size is 256-bits. Each frame in the cache is one fetch packet.

Cache Usage of CPU Addresses

[0029]   Figure 2-3 shows how the cache uses the fetch packet address from the CPU:

5-bit Fetch Packet Alignment: The 5 LSBs of address are assumed to be zero because all program fetch requests are aligned on fetch packet boundaries (8 words or 32 bytes).
11-bit Tag Block Offset: Because the cache is direct mapped, any external address maps to only one of the 2K frame. Any two fetch packets that are an even multiple of 64K byte addresses apart map to the same frame. Thus, bits 15:5 of the CPU address create the 11-bit block offset that determines which of the 2K frames any particular

fetch packet maps to.

10-bit Tag: The cache assumes an a maximum external address space of 64Mbytes (from 00000000-03FFFFFFF). Thus, bits 25:16 correspond to the tag that determines the original location of the fetch packet in external memory space. The cache also has a separate 2KxII tag RAM that holds all the tags. This RAM contains the 10 bit tag plus a valid bit which is used in cache flush.

Cache Flush

[0030] A dedicated valid bit in the tag RAM indicates whether contents of that cache frame contains valid data. During a cache flush all these valid bits are cleared to indicate that no cache frames have valid data. Cache flushes only occurs when the transition of the cache from mapped mode to cache enabled mode.

Frame Replacement

[0031] A cache miss is detected when the tag corresponding to the block offset of the fetch packet address requested by the CPU does not correspond to bits 25:16 of the fetch packet address or if the valid bit at that location is clear. If enabled, the cache loads the fetch packet into the corresponding frame, sets the valid bit, sets the tag to bits 25:16 of the requested address, and delivers this fetch packet to the CPU after all 8 instructions are available.

DMA Access to Program Memory

[0032] The DMA can read and write to internal program memory when configured mapped mode. The priority between the CPU and DMA is configured by the PRI bit of the requesting DMA Channel. If the DMA is higher priority (PRI=1), the CPU is halted while the DMA completes all its requests. If the CPU has priority, the DMA accesses are postponed until the CPU stops making requests. In a cache mode, a DMA write is ignored by the Program Memory Controller. In a cache mode, read returns an undefined value. For both DMA reads and writes in cache modes, the DMA is signaled that its request has completed. At reset, the program memory system is in mapped mode. This feature allows the DMA to bootload code into the internal program memory. See Boot Configuration, Reset, Memory Map for more information on bootloading code.

DATA

Overview

[0033] As shown in Figure 4, the Data Memory Controller connects:

The CPU and Direct Memory Access Controller (DMA) to internal data memory and performs the necessary arbitration.
CPU to the external memory interface (EMIF).
The CPU to the on-chip peripherals through the Peripheral Bus Controller.

[0034] The peripheral bus controller performs arbitration between the CPU and DMA for the on-chip peripherals.

Data Memory Access

[0035] The Data Memory Controller services all requests to internal memory as well as all CPU data requests. Figure 6 shows the directions of data flow as well as the master (requester) and slave (resource) relationships between the modules:

CPU requests data reads and writes to:

1. internal data memory.
2. on-chip peripherals through the peripheral bus controller.
3. EMIF.

DMA requests reads and writes to internal data memory.

CPU sends requests to the Data Memory Controller through the two address buses (DA1 and DA2). Store data is transmitted through the CPU data store buses (ST1 and ST2). Load data is received through the CPU data load buses (LD1 and LD2). The CPU data requests are mapped based on address to either the internal data memory,

internal peripheral space through the peripheral bus controller, or the external memory interface. The Data Memory Controller also connects the DMA to the internal data memory and performs CPU/DMA arbitration for the on-chip data RAM.

5 Internal Data Memory Organization

[0036]   The 64K bytes of internal data RAM located from address 80000000 to 8000FFFF is organized as 4 blocks of 8K blocks of 16-bit halfwords. These blocks are organized in an interleave on 16-bit data with each block containing successive half-words. This interleave allows the two data ports and DMA to read neighboring 16-bit data elements
10 without a resource conflict.

TABLE 2. Data Memory Organization

	8n+0 Block		8n+2 Block		8n+4 Block		8n+6 Block	
First Address	80000000	80000001	80000002	80000003	80000004	80000005	80000006	80000007
	80000008	80000009	8000000A	8000000B	8000000C	8000000D	8000000E	8000000F
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	8000FFF0	8000FFF1	8000FFF2	8000FFF3	8000FFF4	8000FFF5	8000FFF6	8000FFF7
Last Address	8000FFF8	8000FFF9	8000FFFA	8000FFFB	8000FFFC	8000FFFD	8000FFFE	8000FFFF

40 Data Alignment

[0037]   Both the CPU and DMA can read and write 8-bit bytes, 16-bit halfwords, and 32-bit words. The data memory controller performs arbitration between the two CPU independently for each 16-bit block basis. The following data alignment restrictions apply:
45

Words: Word are aligned on even four-byte boundaries (word-boundaries). Words always start at a byte address where the two LSBs are 0. A word access requires two adjacent 16-bit wide blocks.
Halfwords: Halfwords are aligned on even two-byte boundaries (halfword boundaries). Halfwords always start at byte addresses where the LSB is 0.
50 Halfword and require a single 16-bit wide block.
Bytes: There are no alignment restrictions related to byte accesses. Although arbitration is done on 16-bit wide blocks, the blocks still have byte enables to support byte-wide accesses. However, a byte access require the entire 16-block the byte address maps to.

55 Dual CPU Accesses to Internal Memory

[0038]   (CPU clock refers to one period of the CPU input clock (CLKOUT1). (Not the device input clock). CPU cycle refers to the CPU processing one pipeline phase. All instructions in an execute packet proceed through one pipeline

phase in one CPU cycle. However, CPU cycles may be extended to multiple CPU clocks by memory stalls. The extension of a CPU cycle to multiple CPU clocks is known as wait-states.)

[0039]   Interleaved memory organization allows two CPU simultaneous memory accesses. In one CPU cycle, two simultaneous accesses to two different internal memory blocks occur without wait-states. Two simultaneous accesses to the same internal memory block stall the entire CPU pipeline for one CPU clock, providing two accesses in two CPU clocks. These rules apply regardless of whether the accesses are loads or stores. Loads and stores from the same execute packet are seen by the Data Memory Controller during the same CPU cycle. Loads and stores from future or previous CPU cycles do not cause waits for the internal data memory accesses in the current cycle. Thus, internal data memory access only causes wait-states when conflicts occur between instructions in the same fetch packet access the same 16-bit wide block. This condition is called an internal memory conflict. Here, the Data Memory Controller stalls the CPU for one additional CPU clock, serializes the accesses, and performs each access separately. In prioritizing the two accesses, any load occurs before any store access. If both accesses are loads or both accesses are stores, the access from DA1 takes precedence over the access from DA2. Table 3 shows what access conditions cause internal memory conflicts given the when the CPU makes two data accesses (on DA1 and DA2).

TABLE 3. Conflicting Internal Memory Accesses

	DA1	byte								Halfword				word	
DA2	2:0	000	001	010	011	100	101	110	111	000	010	100	110	000	110
byte	000	■	■							■				■	
	001	■	■							■				■	
	010			■	■						■			■	
	011			■	■						■			■	
	100					■	■					■			■
	101					■	■					■			■
	110							■	■				■		■
	111							■	■				■		■
half word	000	■	■							■				■	
	010			■	■						■			■	
	100					■	■					■			■
	110							■	■				■		■
word	000	■	■	■	■					■	■			■	
	110					■	■	■	■			■	■		■

Conflicts shown in shaded areas.

DMA Accesses to Internal Memory

[0040]   If a DMA access to internal memory does not require the same 16-bit banks used by any CPU accesses, the DMA operation occurs in the background. Table 3 may be used to determine DMA versus CPU conflicts. Assume that one axis represents the DMA access and the other represents the CPU access from one CPU data port. Then, perform this analysis again for the other data port. If both comparisons yield no conflict, then there is no CPU/DMA internal memory conflict. Here, the CPU access and DMA access occur in the background of each other. If either comparison yields a conflict, then there is a CPU/DMA internal memory conflict. In this case, the priority is resolved by the PRI bit of the DMA Channel as described later herein. If the DMA Channel is configured as higher priority than the CPU (PRI=1), any CPU accesses are postponed until the DMA accesses complete and the CPU incurs a one CPU clock wait state. If both CPU ports and the DMA access the same memory block, the number of wait-states increases to two. If the DMA has multiple consecutive requests to the block required by the CPU, the CPU is held off until all DMA accesses to the necessary blocks complete. In contrast, if the CPU is higher priority (PRI=0), then the DMA access is postponed until the both CPU data ports stop accessing that block.

Data Endianness

[0041]   Two standards for data ordering in byte-addressable microprocessors exist:

Little endian
Big endian

[0042]   The CPU and DMA support a programmable endianness. This endianness is selected by the LENDIAN (Little ENDIAN) pin on the device. LENDIAN=1 and LENDIAN=0 selects little endian and big endian, respectively. This selection applies to both the CPU and the DMA. Byte ordering within word and half-word data resident in memory is identical for little endian and big endian data. Table 4 shows which bits of a data word in memory are loaded into which bits of a destination register for all possible CPU data loads from big or little endian data. The data in memory is assumed to be the same data that is in the register results from the LDW instruction in the first row. Table 5 shows which bits of a register are stored in which bits of a destination memory word for all possible CPU data stores from big and little endian data. The data in the source register is assumed to be the same data that is in the memory results from the STW instruction in the first row.

TABLE 4.

Register Contents After Little Endian or Big Endian Data Loads			
Instruct ion	Address Bits (1:0)	Big Endian Memory Result	Little Endian Register Result
LDW	00	BA987654h	BA987654h
LDH	00	FFFFBA98h	00007654h
LDHU	00	0000BA98h	00007654h
LDH	10	00007654h	FFFFBA98h
LDHU	10	00007654h	0000BA98h
LDB	00	FFFFFFBAh	00000054h
LDBU	00	000000Bah	00000054h
LDB	01	FFFFFF98h	00000076h
LDBU	01	00000098h	00000076h
LDB	10	00000076h	FFFFFF98h
LDBU	10	00000076h	00000098h
LDB	11	00000054h	FFFFFFBAh
LDBU	11	00000054h	000000Bah
Note: The contents of the word in data memory at location "xxxx xx00" is Ba987654h.			

TABLE 5.

Instruct ion	Address Bits (1:0)	Big Endian Memory Result	Little Endian Memory Result
STW 00	00	BA987654h	BA987654h
STH 00	00	76541970h	01127654h
STH 10	10	01127654h	76541970h
STB 00	00	54121970h	01121954h
STB 01	01	01541970h	01125470h
STB 10	10	01125470h	01541970h
STB 11	11	01121954h	54121970h

Memory Contents After Little Endian or Big Endian Data Stores

Note: The contents of the word in data memory at location "xxxx xx00" before the ST instruction executes is 01121970h. The contents of the source register is BA987654h.

Peripheral Bus

[0043] The peripherals described herein are controlled by CPU and DMA through accesses of control registers. The CPU and DMA access these registers through the peripheral data bus. The DMA directly accesses the Peripheral Bus Controller, whereas CPU accesses it through the Data Memory Controller.

Byte and Halfword Access

[0044] The peripheral bus controller converts all peripheral bus accesses to word accesses. However, on read accesses both the CPU and DMA can extract the correct portions of the word to perform byte and halfword accesses properly. However, any side-effects by causing a peripheral control register read will occur regardless of which bytes are read. In contrast, if byte or halfword writes are performed only the values the CPU and DMA only provide correct values in the enabled bytes. The values guaranteed to be correct are shown in Table 5. Thus, undefined results will be written to the non-enabled bytes. If the user is not concerned about the values in the disabled bytes, this is acceptable. Otherwise, one should only access the peripheral registers via word accesses.

TABLE 6.

Access Type	Address Bits (1:0)	Big Endian Register	Little Endian Memory Result
Word	00	XXXXXXXX	XXXXXXXX
Halfword	00	XXXX????	????XXXX
Halfword	10	????XXXX	XXXX????
Byte	00	XX??????	??????XX
Byte	01	??XX????	????XX??
Byte	10	????XX??	??XX????
Byte	11	??????XX	XX??????

Memory Contents After Little Endian or Big Endian Data Stores

X indicates nybbles correctly written; ? indicates nybbles with undefined value after write

CPU Wait States

[0045] An isolated peripheral bus controller accesses from the CPU causes 4 CPU wait states. These wait-states are inserted to allow pipeline registers to break-up the paths between traversing the on-chip distances between the CPU and peripherals as well as for arbitration time. On consecutive accesses, an accesses after the first only require

3 CPU wait-states due to the pipelined nature of the Data Memory Controller's interface to the peripheral bus controller.

CPU/DMA Arbitration

5   **[0046]**   As shown in Figure 6, the Peripheral Bus Controller performs arbitration between the CPU and DMA for these functions. Like internal data access, the PRI bits in the DMA determine the priority between the CPU and the DMA. If a conflict occurs between the CPU (via the Data Memory Controller) the lower priority requester is held off until the higher priority requester completes all accesses to the peripheral bus controller. The peripheral bus is arbitrated as a single resource, thus the lower priority resource is blocked from accessing all peripherals, not just the one accessed
10   by the higher priority requester.

DMA

[0047]   Referring now to Figure 4, which is a block diagram of DMA controller 143 and interconnected memory mapped
15   modules of microprocessor 1. Direct Memory Access (DMA) Controller 143 transfers data between points in the memory map without intervention by the CPU. The DMA allows movement of data to and from internal memory, internal peripherals, or external devices to occur in the background of CPU operation. The DMA has four independent programmable channels allowing four different contexts for DMA operation. Each DMA channel can be independently configured to transfer data elements of different sizes: 8-bit bytes, 16-bit half-words, or 32-bit words. In addition a fifth (auxiliary)
20   channel allows the DMA to service request from a peripheral with address generation capability such as a host port interface. In discussing DMA operations several terms are important:

Read Transfer: The DMA reads the data element from a source location in memory.
Write Transfer: The DMA writes the data element read during a read transfer to its destination location in memory.
25   Element Transfer: The combined read and write transfer for a single data element.
Frame Transfer: Each DMA channel has an independently programmable number of elements per frame. In completing a frame transfer, the DMA moves all elements in a single frame.
Block Transfer: Each DMA channel also has an independently programmable number of frames per block. In completing a block transfer, the DMA moves all frames it has been programmed to move.
30

[0048]   DMA 143 includes the following features, each of which will be described in following paragraphs:

1) Background Operation: The DMA operates independently of the CPU.
2) High Throughput: Elements can be transferred at the CPU clock rate.
35   3) Four Channels: The DMA can keep track of the contexts of four independent block transfers.
4) Auxiliary Channel: This channel allows a peripheral with address generation capability such as host port 150 to make requests into the CPU's memory space.
5) Split Operation: A single channel maybe used to perform both the receive and transmit transfers to a peripheral, effectively acting like two DMA channels without the additional cost.
40   6) Multi-Frame Transfer: Each block transfer can consist of multiple frames of a fixed programmable size.
7) Programmable Priority: Each channel has independently programmable priorities versus the CPU for each of the memory-mapped resources.
8) Programmable Address Generation: Each channel's source and destination address registers can have configurable indexes through memory on each read and write transfer, respectively. The address may remain constant,
45   increment, decrement, or be adjusted by a programmable value. The programmable value allows a different index for the last transfer in a frame and for the preceding transfers.
9) Full-Address 32-Bit Address Range: The DMA can access any point in the memory map (Figure 3A-3B):

a) the on-chip data memory.
50   b) the on-chip program memory when mapped into memory space rather than being utilized as cache.
c) on-chip peripherals.
d) the external memory interface (EMIF).
e) programmable Width Transfers: Each channel can be independently be configured to transfer either 8-bit bytes, 16-bit half-words, or 32-bit words.
55   f) Auto-Initialization: Once a block transfer is complete, a DMA channel may automatically re-initialize itself for the next block transfer.
g) Event Synchronization: Each read, write, or frame transfer may be initiated by selected events.
h) Interrupt Generation: On completion of each frame transfer or of an entire block transfer as well a on various

error conditions, each DMA channel may send an interrupt to the CPU.

[0049]   DMA 143 is controlled and configured by several memory mapped control registers. Table 7 and Table 8 show how the DMA control registers are mapped into CPU 10's memory space. These registers include the DMA Global Control and Data Registers as well as number of independent control registers for each channel. The DMA Global Data registers are usable as selected by channels for a variety of functions, as described in Table 9. Figure 7 illustrates a DMA Global Data Register.

TABLE 7.

DMA Control Registers by Address	
Hex Byte Address	Name
01840000	DMA Channel 0 Primary Control
01840004	DMA Channel 2 Primary Control
01840008	DMA Channel 0 Secondary Control
0184000C	DMA Channel 2 Secondary Control
01840010	DMA Channel 0 Source Address
01840014	DMA Channel 2 Source Address
01840018	DMA Channel 0 Destination Address
0184001C	DMA Channel 2 Destination Address
01840020	DMA Channel 0 Transfer Counter
01840024	DMA Channel 2 Transfer Counter
01840028	DMA Global Data Register 0
0184002C	DMA Global Data Register 1
01840030	DMA Global Data Register 2
01840034	DMA Global Data Register 3
01840038	DMA Global Data Register 4
0184003C	DMA Global Data Register 5
01840040	DMA Channel 1 Primary Control
01840044	DMA Channel 3 Primary Control
01840048	DMA Channel 1 Secondary Control
0184004C	DMA Channel 3 Secondary Control
01840050	DMA Channel 1 Source Address
01840054	DMA Channel 3 Source Address
01840058	DMA Channel 1 Destination Address
0184005C	DMA Channel 3 Destination Address
01840060	DMA Channel 1 Transfer Counter
01840064	DMA Channel 3 Transfer Counter
01840068	DMA Global Data Register 6
0184006C	DMA Global Data Register 7
01840070	DMA Global Control Register

TABLE 8.

DMA Control Registers by Register Name	
Name	Hex Byte Address
DMA Channel 0 Destination Address	01840018
DMA Channel 0 Primary Control	01840000
DMA Channel 0 Secondary Control	01840008
DMA Channel 0 Source Address	01840010
DMA Channel 0 Transfer Counter	01840020
DMA Channel 1 Destination Address	01840058
DMA Channel 1 Primary Control	01840040
DMA Channel 1 Secondary Control	01840048
DMA Channel 1 Source Address	01840050
DMA Channel 1 Transfer Counter	01840060
DMA Channel 2 Destination Address	0184001C
DMA Channel 2 Primary Control	01840004
DMA Channel 2 Secondary Control	0184000C
DMA Channel 2 Source Address	01840014
DMA Channel 2 Transfer Counter	01840024
DMA Channel 3 Destination Address	0184005C
DMA Channel 3 Primary Control	01840044
DMA Channel 3 Secondary Control	0184004C
DMA Channel 3 Source Address	01840054
DMA Channel 3 Transfer Counter	01840064
DMA Global Control Register	01840070
DMA Global Data Register 0	01840028
DMA Global Data Register 1	0184002C
DMA Global Data Register 2	01840030
DMA Global Data Register 3	01840034
DMA Global Data Register 4	01840038
DMA Global Data Register 5	0184003C
DMA Global Data Register 6	01840068
DMA Global Data Register 7	0184006C

5

10

15

20

25

30

35

40

45

50

55

TABLE 9. DMA Global Data Register Uses

DMA Global Data Register	Source Reload	Destination Reload	Count Reload	Index	Split Address
0			■		
1			■		
2				■	
3				■	
4					■
5	■	■			■
6	■	■			■
7	■	■			

[0050]   DMA Channel Control Registers are illustrated in Figure 8 and Figure 9. The DMA Channel Primary (Figure 8) and Secondary Control Register (Figure 9) contain bit fields that control each individual DMA Channel independently. These fields are described in Table 10 and Table 11, respectively.

TABLE 10.

DMA Channel Primary Control Register Bit Field Definitions	
Bitfield	Description
START	START=00b, stop START=01b, start, without auto-initialization START=10b, pause START=11b, start with auto-initialization
STATUS	STATUS=00b, stopped STATUS=01b, running, without auto-initialization STATUS=10b, paused STATUS=11b, running, with auto-initialization
SRC DIR, DST DIR	Source/Destination Address Modification after Element Transfers. (SRC/DST) DIR=00b, no modification. (SRC/DST) DIR=01b, increment by element size in bytes (SRC/DST) DIR=10b, decrement by element size in bytes. (SRC/DST) DIR=11b, adjust using DMA Global Data Register selected by INDEX.
RSYNC, WSYNC	Read transfer/write transfer synchronization. (R/W)SYNC=00000b, no synchronization. (R/W)SYNC=other sets synchronization event
FS	Frame Synchronization FS=0, disable, FS=1, RSYNC event used to synchronize entire frame.

TABLE 10. (continued)

Bitfield	Description
DMA Channel Primary Control Register Bit Field Definitions	
TCINT	Transfer Controller Interrupt . TCINT=0 interrupt disabled TCINT=1 interrupt enabled
ESIZE	Element size ESIZE=00b, 32-bit ESIZE=01b, 16-bit ESIZE=10b, 8-bit ESIZE=11b, reserved
PRI	Priority Mode: DMA v. CPU PRI=0, CPU priority PRI=1, DMA priority
SPLIT	Split channel mode. SPLIT=00b disabled SPLIT=01b, enabled, use DMA Global Data Register 4 as split address. SPLIT=10b, enabled, use DMA Global Data Register 5 as split address. SPLIT=11b, enabled, use DMA Global Data Register 6 as split address.
CNT RELOAD	DMA Channel Transfer Counter Reload for Auto-Initialization and Multi-Frame Transfers CNT RELOAD=0, Reload with DMA Global Data Register 0 CNT RELOAD=1, Reload with DMA Global Data Register 1
INDEX	Selects the DMA Global Data Register to use as a programmable index. INDEX=0, use DMA Global Data Register 2 INDEX=1, use DMA Global Data Register 3
EMOD	Emulation Mode EMOD=0, DMA channel keeps running during an emulation halt EMOD=1, DMA channel paused during an emulation halt
SRC RELOAD DST RELOAD	DMA Channel Source/Destination Address Reload for Auto-Initialization SRC/DST RELOAD=00b, do not reload during auto-initialization. SRC/DST RELOAD=01b, use DMA Global Data Register 5 as reload. SRC/DST RELOAD=10b, use DMA Global Data Register 6 as reload. SRC/DST RELOAD=11b, use DMA Global Data Register 7 as reload.

TABLE 11.

Bit field	Description
DMA Channel Secondary Control Register Bit Fields	
SX COND FRAME COND LAST COND BLOCK COND (R/W)DROP COND SX IE FRAME IE LAST IE BLOCK IE (R/W)DROP IE (R/W)SYNC	DMA Condition. See 0 for description. COND=0, condition not detected COND=1, condition detected DMA Condition Interrupt Enable. See 0 for description. IE=0, associated condition disables DMA channel interrupt IE=1, associated condition enables DMA channel interrupt Read, Write Synchronization Status

TABLE 11. (continued)

DMA Channel Secondary Control Register Bit Fields	
Bit field	Description
STAT DMAC EN	Write 1 to set associated status. STAT=0, synchronization not received STAT=1, synchronization received DMAC Pin Control DMAC EN=000b, DMAC pin held low DMAC EN=001b, DMAC pin held high DMAC EN=010b, DMAC reflects RSYNC STAT DMAC EN=011b, DMAC reflects WSYNC STAT DMAC EN=100b, DMAC reflects FRAME COND DMAC EN=101b, DMAC reflects BLOCK COND DMAC EN=other, reserved
(R/W)SYNC CLR	Read, Write Synchronization Status Clear Read as 0, write 1 to clear associated status.

[0051] Referring again to Figures 3A-3B, aspects of the DMA's use of the memory map will be described in more detail. Requests are sent to one of four resources:

1) External Memory Interface
2) Internal Program Ram
3) Internal Peripheral Bus
4) Internal Data Ram

[0052] The location of source and destination are computed at the beginning for a block transfer. Thus, the source address is assumed to point to the same one of these four spaces through out a block transfer. This constraint also applies to the destination address.

[0053] Each DMA channel may be started independently either manually through direct CPU access or through auto-initialization. In addition, each DMA channel may be stopped or paused independently through direct CPU access.

[0054] Manual Start Operation: To start DMA operation for a particular channel, once all other DMA control registers are written to their desired values, the DMA Channel Control Register should be written to its desired value with START=01b. Writing this value to a DMA channel that has already been started has no effect.

[0055] Pause Operation: Once started, a DMA channel may then be paused by writing START=10b. When paused, the DMA channel completes any the write transfers element transfers whose read transfer requests have completed. Also, if the DMA channel has all necessary read synchronization, one more element additional element transfer will be allowed to complete. Once paused, the value on STATUS is 10b.

[0056] Stop Operation: The DMA may also be stopped by writing START=00b. Here, the DMA channel stops immediately and discards any data held internally from completed read transfers. The actual status of a DMA channel may be observed by reading the START field in the DMA Channel Control register. Once a DMA transfer is complete, unless auto-initialization is enabled, the DMA Channel returns to the stopped state and STATUS=00b.

[0057] Auto-initialization will now be described. The DMA can automatically reinitialize itself after completion of a block transfer. Some of the DMA control registers can be pre-loaded for the next block transfer through reload registers. Selected DMA Global Data registers act as the reload registers. Using this capability some of the parameters of the DMA channel can be set well in advance of the next block transfer. Auto-initialization allows:

[0058] Continuous Operation: Continuous operation allows the CPU a long slack time during which it can reconfigure the DMA for the subsequent transfer. Normally, the CPU would have to reinitialize the DMA immediately after completion of the last write transfer in the current block transfer and before the first read synchronization for the next block transfer. In general, with the reload registers, it can reinitialize these values for the next block transfer anytime after the current

block transfer begins.

[0059] Repetitive Operation: As a special case of continuous operation, once a block transfer completes the DMA repeats the previous block transfer. In this case, the CPU does not pre-load the reload registers with new values for each block transfer. Instead, it only loads them on the first block transfer.

[0060] Enabling Auto-Initialization: By writing START=11b, in the DMA Channel Control Register, auto-initialization is enabled. In this case, after completion of a block transfer, the DMA Channel is restarted and the selected DMA Channel Registers are reloaded. If restarting after a pause, this START must be re-written as 01b for auto-initialization to be enabled.

[0061] The apparatus of DMA Channel Reload Registers will now be described. For auto-initialization, the successive block transfers are assumed to be similar. Thus, the reload values are only selectable for those registers that are modified during a block transfer: the transfer counter and address registers. Thus, the DMA Channel Transfer Counter as well as the DMA Channel Source and Destination Address Registers have associated reload registers, as selected by the associated RELOAD fields in the DMA Channel Primary Control Register (Table 10). The reload registers are stored in particular DMA Global Data registers.

[0062] Note that it is possible to not reload the source or destination address register in auto-initialization mode. This capability allows a register to maintain its value that did not change during block transfer. Thus, a DMA Global Data Register does not have to be dedicated to a value that was static during block transfer. A single channel may use the same value for multiple channel registers. For example, in split mode, the source and destination address maybe the same. Similarly, multiple channels may use the same reload values. For example, two channels may have the same transfer count reload value.

[0063] Upon completion of a block transfer, these registers are reloaded with the associated reload register. Note that in the case of the DMA Channel Transfer Counter Register, reload occurs after the end of each frame transfer, not just after the end of the entire block transfer. The reload value for the DMA Channel Transfer Counter is necessary whenever multi-frame transfers are configured, not just when auto-initialization is enabled.

[0064] As discussed earlier, the DMA may allow read transfers to get ahead of write transfers and provides the necessary buffering to facilitate this capability. To support this, the necessary reload at the end of blocks and frames occurs independently for the read (source) and write (destination) portions of the DMA Channel. Similarly, in the case of split channel operation, described later, the source and destination address are independently reloaded when the associated transmit or receive element transfers, respectively, complete a block transfer.

[0065] The DMA Channel Transfer Counter Reload can only be re-written by the user after the next to last frame in the current block transfer completes. Otherwise, the new reload values would affect subsequent frame boundaries in the current block transfer. However, if the frame size is the same for the current and next block transfers, this restriction is not relevant. A fuller explanation of the DMA Channel Transfer Counter is provided in later paragraphs.

[0066] Figure 10 illustrates a DMA Channel Transfer Counter. The DMA Channel Transfer Counter contains bit fields that represent the number of frames and the number of elements per frame to be transferred. Figure 11 shows how a DMA Global Data Register can be used as the reload value for the transfer counter.

[0067] FRAME COUNT: This 16-bit unsigned value sets the total number of frames in the block transfer. Thus, if a block transfer contains a single frame, this value should be set to its default of 1. The maximum number of frames per block transfer is 65535. This counter is decremented upon the completion of the last read transfer in a frame transfer. Once the last frame is transferred, the entire counter is reloaded with the DMA Global Data Register selected by the CNT RELOAD field in the DMA Channel Primary Control Register. Also note that initial values of 0 and 1 to FRAME COUNT have the same effect and a single frame will be transferred.

[0068] ELEMENT COUNT: This 16-bit unsigned value sets the number of elements per frame. This counter is decremented after the read transfer of each element. The maximum number of elements per frame transfer is 65535. Once the last element in each frame, is reached, ELEMENT COUNT is reloaded with the 16 LSBs of the DMA Global Data Register selected by the CNT RELOAD field in the DMA Channel Primary Control Register. This reloading is unaffected by auto-initialization mode. Before block transfer begins, the counter and selected DMA Global Data Register must be loaded with the same 16 LSBs to assure that the first and remaining frames have the same number of elements per frame. Also, in any multi-frame transfer, a reload value must be specified, not just when auto-initialization is enabled. If element count is initialized as 0, operation is undefined.

[0069] Synchronization will now be described. Synchronization allows DMA transfers to be triggered by events such as interrupts from internal peripherals or external pins. Three types of synchronization may be enabled for each channel:

    1) Read Synchronization: Each read transfer waits for the selected event to occur before proceeding.

    2) Write Synchronization: Each write transfer waits for the selected event to occur before proceeding.

    3) Frame Synchronization: Each frame transfer waits for the selected event to occur before proceeding.

[0070] Selection of Synchronization Events: The events are selected by the RSYNC and WSYNC fields in the DMA

Channel Primary Control Register. (Figure 8) If FS=1 in this register, then the event selected by RSYNC enables an entire frame. Up to 31 events are available. If the value of these fields is set to 00000b then no synchronization is necessary. In this case, the read, write, or frame transfers occur as soon as the resource is available to that channel. The association between values in these fields to events is shown in Table 12.

TABLE 12.

Synchronization Events		
Event Number (Binary)	Event Acronym	Event Description
00000	None	no synchronization
00001	TINT0	Timer 0 Interrupt
00010	TINT1	Timer 1 Interrupt
00011	SD_INT	EMIF SDRAM Timer Interrupt
00100	EXT_INT4	External Interrupt Pin 4 INT (4)
00101	EXT_INT5	External Interrupt Pin 5 INT (5)
00110	EXT_INT6	External Interrupt Pin 6 INT (6)
00111	EXT_INT7	External Interrupt Pin 7 INT (7)
01000	DMA_INT0	DMA Channel 0 Interrupt
01001	DMA_INT1	DMA Channel 1 Interrupt
01010	DMA_INT2	DMA Channel 2 Interrupt
01011	DMA_INT3	DMA Channel 3 Interrupt
01100	XEVT0	MCSP 0 Transmit Event
01101	REVT0	MCSP 0 Receive Event
01110	XEVT1	MCSP 1 Transmit Event
01111	REVT1	MCSP 1 Receive Event
10000	DSPINT	Host Port 150Host to DSP Interrupt
Other	Reserved	
Note: in Table 12, MCSP refers to multichannel serial port 120, described later herein.		

[0071] The DMA Channel Secondary Control Register (Table 11) contains STAT and CLR fields for read and write synchronization events.

[0072] Latching of DMA Synchronization Events: An inactive to active transition of the selected event is latched by each DMA channel. The occurrence of this transition causes the associated STAT field to be set in the DMA Channel Secondary Control register. Note that if no synchronization is selected the STAT bit is always read as 1. Also, note that a single event can trigger multiple actions.

[0073] User Clearing and Setting of Events: By clearing pending events before starting a block transfer the DMA Channel can be forced to wait for the next event. Conversely, by setting events before starting a block transfer the synchronization events necessary for the first element transfer can be forced. Events (and thus the related STAT bit may be clear or set) by writing 1 to the a corresponding CLR or STAT field, respectively. Note that writing a zero to either of these bits has no effect. Also, the CLR bits are always read as 0 and have no associated storage. Separate bits for setting or clearing are provided to allow clearing of some bits without setting others and vice-versa. Note that user manipulation of events has priority over any simultaneous automated setting or clearing of events.

[0074] Automated Event Clearing will now be described. The latched STAT for each synchronizing event is only cleared when any action associated with that event completes. Events are cleared as quickly as possible to reduce the minimum time between synchronizing events. This capability effectively increases the throughput at which events can be recognized. This is described in detail for each type of synchronization below:

[0075] Clearing Read Synchronization Condition: The latched condition for read synchronization is cleared when the DMA completes the request for the associated read transfer.

[0076] Clearing Write Synchronization Condition: The latched condition for write synchronization is cleared when

the DMA completes the request for the associated write transfer.

[0077] Clearing Frame Synchronization Condition: Frame synchronization is clears the RSYNC STAT field when the DMA completes the request for the first read transfer in the new frame.

[0078] Address Generation will now be described. For each channel, the DMA performs address computation for each read transfer and write transfer. The DMA allows creation of a variety of data structures. For example, the DMA can traverse an array striding through every nth element. Also, it can be programmed to effectively treat the various elements in a frame as coming from separate sources and group each source's data together.

[0079] Figure 12 and Figure 13 illustrate the DMA Channel Source Address and Destination Address Registers, which hold the addresses for the next read transfer and write transfer, respectively.

[0080] As shown in Figure 8, the SRC DIR and DST DIR fields can set the index to increment, to decrement, or to not effect the DMA Channel Source and Destination Address Registers, respectively. By default, these values are set to 00b to disable any incrementing or decrementing. If incrementing or decrementing is enabled, then address adjustment amount is by the element size in bytes. For example, if the source address is set to increment and 16-bit half-words are being transferred, then the address is incremented by 2 after each read transfer.

[0081] Address Adjustment with the DMA Channel Index Registers: As shown in Table 10, the SRC DIR and DST DIR field can independently allow for selecting a particular DMA Global Data Register, illustrated in Figure 14, to determine the address adjustment. The particular DMA Global Data Register is selected via INDEX field in the DMA Channel Primary Control Register. Unlike basic address adjustment, this mode allows different adjustment amount depending on whether or not the element transfer is the last in the current frame. The normal adjustment value (ELEMENT INDEX) is contained in the 16 LSBs of the selected DMA Global Data Register. The adjustment value (FRAME INDEX) for the end of the frame; is determined by the 16 MSBs of the selected DMA Global Data Register. Both these fields contain signed 16-bit values. Thus, the index amounts can range from - 32768 to 32767.

[0082] These fields affect address adjustment as follows:

1) ELEMENT INDEX: For all except the last transfer in a frame, ELEMENT INDEX determines the amount to be added to the DMA Channel Source for Destination Address Register as selected by the SRC DIR or DST DIR field after each read or write transfer, respectively

2) FRAME INDEX: If the read or write transfer is the last in a frame, FRAME INDEX (and not the ELEMENT INDEX) field is used for address adjustment. This occurs in both single frame and multi-frame transfers.

[0083] Element Size, Alignment, and Endianness: Using the ESIZE field in the DMA Channel Control Register, the user may configure the DMA to transfer 8-bit bytes, 16-bit halfwords, or 32-bit words on each transfer. The following registers and bit fields must be loaded with properly aligned values:

1) DMA Channel Source and Destination Address Registers and any associated reload registers.
2) ELEMENT INDEX
3) FRAME INDEX

[0084] In the case of word transfers, these registers must contain values that are multiples of 4, thus aligned on a word address. In the case of half-word transfers they must be multiples of 2, thus aligned on a half-word address. If unaligned values are loaded, operation is undefined. There is no alignment restriction for byte transfers. All accesses to program memory must be 32-bits in width. Also, be aware of the endianness when trying to access a particular 8-bit or 16-bit field within a 32-bit register. For example, in little endian, an address ending in 00b select the LSbyte whereas llb selects the LSbyte in big endian.

[0085] An example using frame index to reload addresses will now be described. In an auto-initialized, single frame block transfer the FRAME index can be used in place of a reload register to re-compute the next address. For example, consider a single frame transfer where 10 bytes are to be moved from a static external address to alternating locations (skip one byte):

SRC DIR=00b, static source address.
DST DIR=11b, programmable index value
ELEMENT INDEX=10b, 2 byte destination stride
FRAME INDEX=9x2=18=10010b, correct by -18 byte locations to restart destination at same place.

[0086] An example of transferring a large single block will now be described. The ELEMENT COUNT and FRAME COUNT can be used in conjunction to effectively allow single frame block transfers of greater than 65535 in size. Here, the product of the element count and frame count can form a larger effective element count. The following must be performed:

1) If the address is set to be adjusted using a programmable value (DIR=11b), the FRAME INDEX must equal the ELEMENT INDEX if the address adjustment is determined by a DMA Global Data Register. This applies to both source and destination addresses. If the address is not set to be adjusted by a programmable value, this constraint does not apply because by default the same address adjustment occurs at element and frame boundaries.

2) Frame synchronization must be disabled (FS=0 in the DMA Channel Primary Control Register). This prevents requirements for synchronization in the middle of the large block.

3) The number of elements in the first frame is Ei. The number of element in successive frames is $((F-1) \times Er)$ The effective element count will be $((F-1) \times Er) + Ei$. Where:

F = The initial value of the FRAME COUNT
Er = ELEMENT COUNT Reload value
Ei = initial value of the ELEMENT COUNT

[0087] Thus, to transfer 128K + 1 elements, one could set the F=5, Er=32K, and Ei=1.

[0088] An example of sorting will now be described. To have transfers located in memory by ordinal location within a frame (i.e. the first transfer of the first frame followed by the first transfer of the second frame):

1) ELEMENT INDEX should be set to: F x S.
2) FRAME INDEX be set to: $-(((E-1) \times F)-1) \times S$, where

E= the initial value of ELEMENT COUNT (the number of elements per frame) as well as the ELEMENT COUNT RELOAD.
F = the initial value of FRAME COUNT (the total number of frames).
S = the element size in bytes.

[0089] Consider a transfer with three frames (F=3) of four half-word elements each (E=4, S=2). Thus, ELEMENT INDEX=3x2=6 and FRAME INDEX=$-(((4-1)\times3)-1)\times2=-16$. Assume that the source address is not modified and the destination increments starting at 0x80000000.

[0090] Table 13 and Table 14 show how this sorting works for this example.

TABLE 13.

Sorting Example in Order of DMA Transfers			
Frame	Element	Address	Post Adjustment
0	0	0x80000 000	+6
0	1	0x80000 006	+6
0	2	0x80000 00C	+6
0	3	0x80000 012	-16
1	0	0x80000 002	+6
1	1	0x80000 008	+6
1	2	0x80000 00E	+6
1	3	0x80000 014	-16
2	0	0x80000 004	+6
2	1	0x80000 00A	+6
2	2	0x80000 010	+6
2	3	0x80000 016	-16

TABLE 14.

Sorting Grouping Ordered By Address		
Frame	Element	Address
0	0	0x80000000
1	0	0x80000002
2	0	0x80000004
0	1	0x80000006
1	1	0x80000008
2	1	0x8000000A
0	2	0x8000000C
1	2	0x8000000E
2	2	0x80000010
0	3	0x80000012
1	3	0x80000014
2	3	0x80000016

[0091]   Split Channel operation will now be described. Split channel operation allows a single DMA channel to provide the capability of two channels to service both the input (receive) and output (transmit) streams from an external or internal peripheral with a fixed address.

[0092]   Figure 15 illustrates a DMA Global Data Register used for split addresses. The DMA Global Data Register selected by the SPLIT field in the DMA Primary Control Register determines the address of the peripheral that is to be access for split transfer:

[0093]   Split Source Address: This address is the source for the input stream to the 'Processor 1. The selected DMA Global Control Register contains this split source address.

[0094]   Split Destination Address: This address is the destination for the output data stream from 'Processor 1. The split destination address is assumed to be one word address (4 bytes addresses) greater than the split source address.

[0095]   Notice that the 3 LSBs are fixed at 0. The 2 LSBs are fixed at zero to force alignment at a word address. The third LSBs is 0, because the split source address is assumed to be on an even word boundary. Thus, the split destination address is assumed to be on an odd word boundary. These relationships hold regardless of the width of the transfer. Internal peripherals will conform to this convention. For external peripherals, design address decode to appropriately adhere to this convention.

[0096]   Split DMA Operation will now be described. Split operation consists of transmit element transfers and receive element transfers. In turn, these each consist of a read and a write transfer:

1) Transmit Element Transfer

  a) Transmit Read Transfer: Data is read from the DMA Channel Source Address. The Source Address is then adjusted as configured. This event is not synchronized.
  b) Transmit Write Transfer: Data from the transmit read transfer is written to the split destination address. This event is synchronized as indicated by the WSYNC field. The transfer count is then decremented. The DMA channel internally keeps track of the number of pending receive transfers.

2) Receive Element Transfer

  a) Receive Read Transfer: Data is read from the split source address. This event is synchronized as indicated by the RSYNC field.
  b) Receive Write Transfer: Data from the receive read transfer is written to the Destination Address. The destination address is then adjusted as configured. This event is not synchronized.

[0097]   Note, since only a single Element Count and Frame Count exists per channel, the ELEMENT COUNT and

the FRAME COUNT are the same for both the received and the transmitted data. For split operation to work properly, both the RSYNC and WSYNC fields must be set to synchronization events. Also, frame synchronization must be disabled in split mode.

[0098]   For all transfers the above sequence is maintained. However, the transmit transfers do not have to wait for all previous receive element transfers to complete before proceeding. Therefore, it is possible for the transmit stream to get ahead of the receive stream. The DMA Channel Transfer counter decrements (or reinitialize) after the associated transmit transfer completes. However, re-initialization of the source address register occurs after all transmit element transfers complete. This configuration works as long as transmit transfers do not get eight or more transfers ahead of the receive transfers. In that case, transmit element transfers will be stopped, possibly causing missing of synchronization events. For cases where receive or transmit element transfers are within seven or less transfers of the other, the DMA channel maintains this information as internal status.

[0099]   Resource Arbitration and Priority Configuration will now be described. Priority decides which of competing requesters have control of a resource with multiple requests. The requesters include:

1) the DMA Channels
2) the CPU's program and data accesses.

[0100]   The resources include:

1) internal Data Memory including each interleave of internal data memory.
2) the internal peripheral registers which are accessed through the peripheral bus.
3) internal program memory.
4) the External Memory Interface (EMIF).

[0101]   Two aspects of priority are programmable:

1) DMA versus CPU Priority Each DMA channel may independently be configured in high priority mode by setting the PRI bit in the associated DMA Channel Control Register. The AUXPRI field in the DMA Global Control Register allows same feature for the auxiliary channel. When in high priority mode, the associated channel's requests are sent to the appropriate resource with a signal indicating the high priority status. By default all these fields are 0, disabling the high priority mode. Each resource can use this signal in its own priority scheme for resolving conflicts. Refer to the documentation for the particular resource for how it utilizes this signal.
2) Priority Between DMA Channels: The DMA has a fixed priority scheme with channel 0 having highest priority and channel 3 having lowest priority. The auxiliary channel may be given a priority anywhere within this hierarchy.

[0102]   Figure 16 illustrates the DMA Global Control Register which specifies Priority Between Channels. The fields in the DMA Global Control Registers affect all DMA channels and are described in Table 15. The fields in this register will be referred to in the following sub-sections.

TABLE 15.

DMA Global Control Register	
Bit field	Description
CH PRI	DMA Channel Priority CH PRI=0000b, fixed channel priority mode auxiliary channel 1st highest priority CH PRI=0001b, fixed channel priority mode auxiliary channel 2nd highest priority CH PRI=0010b, fixed channel priority mode auxiliary channel 3rd highest priority CH PRI=0011b, fixed channel priority mode auxiliary channel 4th highest priority CH PRI=0100b, fixed channel priority mode auxiliary channel 5th highest priority CH PRI=other, reserved
AUXPRI	Auxiliary Channel Priority Mode AUXPRI=0, CPU priority AUXPRI=1, DMA priority

[0103]   The priority between DMA channels determines which DMA channel will perform a read or write transfer first, given that two or more channels are ready to perform transfers.

[0104]   The priority of the auxiliary channel is configurable by programming the CH PRI field in the DMA Global Control

Register. By default, CH PRI=0000b at reset. This sets the auxiliary channel as highest priority, followed by channel 0, followed by channel 1, followed by channel 2, with channel 3 having lowest priority.

[0105]   Arbitration between channels occurs independently for read and write transfers every CPU clock cycle. Any channel that is in the process of waiting for synchronization of any kind may lose control of the DMA to a lower priority channel. Once that synchronization is received, that channel may regain control of the DMA from a lower priority channel. This rule is applied independently to the transmit and receive portions of a split mode transfer. The transmit portion has higher priority than the receive portion.

[0106]   If multiple DMA channels and the CPU are contending for a resource, the arbitration for which DMA channel has priority occurs logically first. Then arbitration between the highest priority DMA channel and the CPU occurs. Normally, if a channel is lower priority than the CPU, all lower priority channels should also be lower priority than the CPU. Similarly, if a channel is higher priority than the CPU, all higher priority channels should also be higher priority than the CPU. This arbitration of DMA versus CPU contention is decided by each particular resource. Refer to that resource's documentation for a full explanation. Note that the PRI field should only be modified when that channel is paused or stopped.

[0107]   A higher priority channel will gain control of the DMA from a lower priority channel once it has received the necessary read synchronization. In switching channels, the current channel allows all data from requested reads to complete. Then the DMA determines which higher priority channel will gain control of the DMA controller read operation. That channel then starts its read operation. Simultaneously, write transfers from the previous channel are allowed to complete.

[0108]   If multiple DMA channels are contending for the same resource for reads AND writes, then the higher priority channel wins. For example, if channel 0 wants to read from the EMIF and channel 1 wants to write to the EMIF, then the channel 0 reads occur first. If one channel is requesting both reads and writes from/to the same resource, then the writes happen first.

[0109]   Methods for DMA channel condition determination will now be described. Several conditions are available to inform the user of significant milestones or potential problems in DMA channel operation. These events (indicated by the COND bit fields) are held in the DMA Channel Secondary Control Register.

[0110]   This register also provides the means to enable these events to trigger the DMA to CPU interrupt for that channel x through the corresponding interrupt enable (IE) bit fields. If a COND bit and its corresponding IE bit are set then that condition is enabled to contribute to the status of the interrupt signal the from the associated DMA Channel to the CPU. If the TCINT bit in the DMA Channel x Control register is set, the logical OR of all enabled conditions forms the DMA_INTx signal. Otherwise, the DMA_INTx remains inactive. This logic is shown in Figure 17. If selected by the interrupt selector, a low to high on that DMA_INT will cause an interrupt condition to be latched by the CPU.

[0111]   The SX COND, WDROP, and RDROP bits in the DMA Channel Secondary Control Register are treated as warning conditions. If these conditions are enabled and active, then they move the DMA channel from the running to the pause state, regardless of the value of the TCINT bit.

[0112]   If a COND bit's associated IE bit is set, that COND bit may only be cleared by a user write of 0. Otherwise, that COND bit may be automatically cleared as indicated in the following section. A user write of 1 to a COND bit has no effect. Thus, manually forcing one of the conditions cannot be done.

[0113]   Most values in this register are cleared at reset. The one exception is the interrupt enable for the block transfer complete event (BLOCK IE), which is set at reset. Thus, by default, the block transfer complete condition is the only condition that could contribute to the CPU interrupt. Other conditions can be enabled by setting the associated IE bit.

[0114]   Table 16 describes each of the conditions in the DMA Channel Secondary Control Register. Depending on the system application, these conditions may represent errors. As a note, the last frame condition can be used to change the reload register values for auto-initialization. The frame index and element count reload are used every frame. Thus, wait until all but the last frame transfer in a block transfer completes to change these values. Otherwise, the current (rather than the next) block transfer will be affected.

TABLE 16.

DMA Channel Condition Descriptions				
Bit field	Event	Occurs if...	COND Cleared By	
Name			If IE Enabled	Otherwise
SX	Split Transmit Overrun Receive	The split operation is enabled and transmit element transfers get seven or more element transfers a ahead of receive element transfers.	A user write of 0 to COND transfers.	
FRAM E	Frame Complete	After the last write transfer in each frame is written to memory.	A user write of 0 to COND.	Two CPU clocks later.
LAST	Last Frame	After all counter adjustments for the next to last frame in a block transfer complete.	A user write of 0 to COND.	Two CPU clocks later.
WDRO P RDRO P	Dropped Read/Write Synchron ization	A subsequent synchronization event occurs before the last one is cleared.	A user write of 0 to COND.	
BLOC K	Block Transfer Complete	After the last write transfer in a block transfer is written to memory.	A user write of 0 to COND.	Two CPU clocks later.

[0115] Figure 18 shows the internal data movement paths of the DMA controller including data buses and internal holding registers.

[0116] Each DMA channel can independently select one of four sources and destinations:

1) EMIF 103
2) Internal Program Memory 23
3) Internal Data Memory 22
4) Internal Peripheral Bus 110

[0117] Thus, read and write buses are provided from each interface to the DMA controller.

[0118] The auxiliary channel also has read and write buses. However, as the auxiliary channel provides address generation for the DMA the naming convention of its buses differ. For example, data writes from the auxiliary channel through the DMA are performed through the Auxiliary Write Bus. Similarly, data reads from the auxiliary channel through the DMA are performed through the Auxiliary Read Bus.

[0119] An aspect of the present invention is a 9-deep DMA FIFO 300 holding path that is provided to facilitate bursting to high performances memories including internal program and data memory as well as external synchronous DRAM (SDRAM) or synchronous burst SRAM (SBSRAM). When combined with a channel's holding registers 310a, 310b, 310c or 310d, this effectively becomes an 11-deep FIFO.

[0120] There are actually three components to the FIFO:

1) An address FIFO 320 which stores the two LSBs of the read address and the end_of_frame and end_of_block status at the time of read adv.

2) A 36-bit wide data FIFO 300 which stores the 32-bit read data word along with the data coming out of the address FIFO due to ack,s.

3) A two-deep intermediate stage 310a-310d that stores data coming out of the address stage before it goes to the data FIFO which is required because of the two-cycle ack to read data latency.

[0121] During the ten read adv cycles before any write adv's, read ack's begin. These ack's start removing data from the address FIFO 320 and placing them in the data FIFO 300 along with read data. Thus, the data items are distributed across the three stages. There is counting logic 340 that tracks the number of items stored in each stage.

[0122] At any one time only one channel controls the FIFO. For a channel to gain control of the FIFO, the following conditions must all apply:

1) The channel has no read or write synchronization enabled. Since split mode requires read and write synchronization, the FIFO is not used by a channel in split mode. Note that if only frame synchronization is enabled then the FIFO may still be used by that channel.

2) The channel is running.

3) The FIFO is void of data from any other channel.

4) The channel is the highest priority channel of those that meet the above three conditions.

[0123] The third restriction minimizes "head-of-line" blocking. Head-of-line blocking occurs when a DMA request of higher priority waits for a series of lower priority requests to come in before issuing its first request. If a higher priority channel requests control of the DMA controller from a lower priority channel, only the last request of the previous channel has to complete. After that, the higher priority channel completes its requests through its holding registers. The holding registers do not allow as high a throughput through the DMA controller. In the gaps, the lower priority channel begins no more read transfers but is allowed to flush the FIFO by completing its write transfers. As the higher priority channel is not yet in control of the FIFO, there will be gaps in its access where the lower priority channel may drain its transfer from the FIFO. Once the FIFO is clear, if the higher priority channel has not stopped, it gains control of the FIFO.

[0124] The DMA FIFO has two purposes:

1) Increased Performance

2) Decreased Arbitration Latency

[0125] Increased Performance: The FIFO allows read transfers to get ahead of write transfers. This feature minimizes penalties for variations in available transfer bandwidth at either end of the element transfer. Thus, the DMA can capitalize on separate windows of opportunity at the read and write portion of the element transfer. If the requesting DMA channel is using the FIFO, the resources are capable of sustaining read or write accesses at the CPU clock cycle rate. However, there may be some latency in performing the first access. The handshaking between the resource and the DMA Controller controls the rate of consecutive requests and the latency of received read transfer data.

[0126] To sustain read and write accesses at the CPU clock rate, the FIFO cannot be filled. To avoid filling the FIFO, data must begin being written out of the FIFO before read requests place data in the last empty FIFO location. From the peripheral handshaking operation, we have determined that a fifo depth of eleven words is required. Thus, at any point in time, the DMA may have up to eleven read transfers in the FIFO queued for their write transfers to complete.

[0127] Decreased Arbitration Latency versus the CPU: To capture read data from any pending requests for a particular resource. For example, consider the situation where the DMA is reading data from pipelined external memory such as SDRAM or SBSRAM to internal data memory. Assume the CPU is given higher priority over the DMA channel making requests and that it makes a competing program fetch request of the EMIF. Also, assume that simultaneously, the CPU is accessing all banks of internal memory, blocking out the DMA. In this case, the FIFO allows the pending DMA accesses to complete and the program fetch to proceed. Due to the pipelined request structure of the DMA, at any one point in time the DMA may have up to eleven pending read transfer requests whose data has not yet arrived. Once eleven requests are outstanding, the DMA stops making subsequent read transfer requests.

[0128] Each channel has dedicated internal holding registers. If a DMA channel is transferring data through its holding registers rather than the internal FIFO, read transfers are issued consecutively. Once a read transfer request has been initiated, no subsequent read transfer is started until the read data has arrived within the holding register. Depending whether the DMA controller is in split mode or not additional restrictions can apply:

1) Split Mode: The two registers serve as separate transmit and receive data stream holding registers for split mode. For either the transmit or receive read transfer, no subsequent read transfer request is issued until the associated write transfer request completes.

2) Non-Split Mode: However, when not in split mode, once the data arrives a subsequent read transfer may be issued without waiting for the associated write transfer to complete. However, because there are two holding registers, read transfers may only get one ahead of write transfers.

[0129] Using the described structure, the DMA can perform element transfers with single cycle throughput, if it accesses separate resources for the read transfer and write transfer and both these resources have single-cycle throughput. An example would be an unsynchronized block transfer from single-cycle external SBSRAM to internal data memory without any competition from either other channels or the CPU. The DMA performance can be limited by:

1) The throughput and latency of the resources it requests.
2) Waiting for read, write, or frame synchronization.
3) Contention for resources for other DMA channels.

[0130] Referring again to Figure 4, DMA Action Complete Pins, DMAC (0-3), will now be described. The DMA Action Complete pins provide a method of feedback to external logic generating an event for each channel (DMAC0-DMAC3). As decided by the DMAC EN bit field in the DMA Channel Secondary Control register, this pin can reflect the status of RSYNC STAT, WSYNC STAT, BLOCK COND, or FRAME COND, or be treated as a high or low general purpose output. If DMAC reflects RSYNC STAT or WSYNC STAT, externally, once a synchronization event has been recognized DMAC will transition from low to high. Once that same event has been serviced (as indicated by the status bit being cleared), DMAC transitions from high to low. Before being sent off chip the DMAC signals are synchronized by CLKOUT2 (½ the CPU clock rate). The active period of these signals is guaranteed to be a minimum of 2 CLKOUT2 periods wide. Also, even if before synchronization the pulses are only 1 CPU clock period wide, a minimum 2 CLKOUT period active-high pulse occurs the DMAC pin.

[0131] Referring again to Figure 1A, during debug using an emulator in test system 51, the CPU may be halted on an execute packet boundary for single stepping, benchmarking, profiling, or other debug uses. The user may configure whether the DMA pauses during this time or continues running. This function is performed by setting the EMOD bit in the DMA Primary Control register to 0 or 1, respectively. If paused, the STATUS field will reflect the pause state of the channel. The auxiliary channel continues running during an emulation halt. European Patent application No. 97310468.0 describes emulation of microprocessor 1 in complete detail, and is incorporated herein by reference.

HPI

Overview

[0132] As depicted in Figure 4, the host port interface (HPI) is a parallel port through which a host processor can have direct access the CPU's memory space. The host device is master of the interface, therefore increasing its ease of access. The host and CPU can exchange information via internal or external memory. In addition, the host has direct access to memory mapped peripherals. Connectivity to the CPU's memory space is provided through the DMA controller. Dedicated address and data registers not accessible to the CPU connect the HPI to the DMA Auxiliary Channel which in turn connects the HPI to the CPU's memory space. To configure the interface, the HPI and CPU can access the HPI Control Register (HPIC). The host can access the host address register (HPIA) and host data register (HPID) as well as the HPIC using the external data and interface control signals (Figure 19).

[0133] Figure 19 shows a simplified diagram of host interface to the HPI. The HPI provides 32-bit data to the CPU with an economical 16-bit external interface by automatically combining successive 16-bit transfers. When the host device transfers data through HPID, the DMA Auxiliary Channel accesses the CPU's address space. The HPI supports high speed, back-to-back host accesses. In the absence of contention for memory mapped-resources the HPI can transfer one 16-bit value every 4 CPU clocks. Thus, the DMA auxiliary channel can perform one 32-bit access every 8 CPU clocks.

[0134] The external HPI interface consists of the 16-bit HPI data bus and control signals that configure and control the interface. The interface can connect to a variety of host devices with little or no additional logic necessary.

[0135] The 16-bit data bus (HD0--HD15) exchanges information with the host. Because of the 32-bit word structure of the chip architecture, all transfers with a host consist of two consecutive 16-bit halfwords. On host data (HPID) write accesses, the HBE-1:0 byte enables select which bytes in a 32-bit word should be written. HPIA, HPIC accesses, as well as HPID read accesses are performed as 32-bit accesses and the byte enables are not used. The dedicated HHWIL pin indicates whether the first or second halfword is being transferred. An internal control register bit determines whether the first or second halfword is placed into the most significant halfword of a word The host must not break the first halfword/second halfword (HHWIL low/high) sequence of an ongoing HPI access.

[0136] The two data strobes (HDS1- and HDS2-), the read/write select (HR/W-), and the address strobe (HAS-)

enable the HPI to interface to a variety of industry-standard host devices with little or no additional logic required. The HPI can easily interface to hosts with multiplexed or dedicated address/data bus, one data strobe and a read/write strobe, or two separate strobes for read and write.

[0137]   The HCNTL1:0 control inputs indicate which HPI register is accessed. Using these inputs, the host can specify an access to the HPIA (which serves as the pointer into CPU's memory space), HPIC, or HPID. These inputs, along with HHWIL, are commonly driven by host address bus bits or a function of these bits. By writing to the HPIC, the host can interrupt the CPU, whereas the CPU can activate HINT- interrupt output to the host.

[0138]   The host can access HPID with an optional automatic address increment of HPIA. This feature facilitates reading or writing to sequential word locations. In addition, auto-increment HPID reads prefetch the data at the auto-incremented access to reduce latency on the subsequent host read request. The HPI ready pin (HRDY-) allows insertion of host wait states. Wait-states may be necessary depending on latency to the point in the memory map accessed via the HPI as well as on the rate of host access. The rate of host access may force not-ready conditions if the host attempts to access the host port before any previous HPID write access or prefetched HPID read access completes. In this case, the HPI simply holds off the host via HRDY-. HRDY- provides a convenient way to automatically (no software handshake needed) adjust the host access rate to the rate of data delivery from the DMA auxiliary channel. In the cases of hardware systems that cannot take advantage of HRDY- pin, a HRDY bit in the HPIC is available for use as a software handshake.

HPI Signal Description

[0139]   The external HPI interface signals implement a flexible interface to a variety of types of host devices. Table 17 lists the HPI pins and their functions. The remainder of this section discusses the pins in detail.

TABLE 17.

HPI External Interface Signals				
Signal Name	Signal Type (Input/ Output/HI-Z)	Sig nal Count	Host Connection	Signal Function
HD15:0	I/O/Z	16	Data Bus	
HCNTL1 :0	I	2	Address or control lines	Controls HPI access type.
HHWIL	I	1	Address or control lines	Halfword identification input.
HAS-	I	1	Address latch enable (ALE) or Address strobe or unused (tied high)ÁÁÁ	Differentiates address versus data values on multiplexed address/ data host.
HBE-1:0	I	2	Byte enables	Data write byte enables
HR/W-	I	1	Read/Write strobe, address line, or multiplexed address/ data	Read/Write select
HCS-	I	1	Address or control lines	Data strobe inputs.
HDS1-HDS2-	I	2	Read strobe and write strobe or data strobe	Data strobe inputs.
HRDY-	O	1	Asynchronous ready	Ready status of current HPI access
HINT-	O	1	Host interrupt input.	Interrupt signal to Host

Data Bus: HD15:0

[0140]   HD15:0 is a parallel bi-directional 3-state data bus. HD is placed in high-impedance when not performing an HPI read access.

Access Control Select: HCNTL1:0

[0141] HCNTL1:0 indicate which internal HPI register is being. The states of these two pins select access to the HPI address (HPIA), HPI data (HPID), or HPI control (HPIC) registers. The HPIA register serves as the pointer into HPI memory, the HPIC contains control and status bits for the transfers, and the HPID contains the actual data transferred. Additionally, the HPID register can be accessed with an optional automatic address increment. Table 18 describes the HCNTL0/1 bit functions.

TABLE 18.

HPI Input Control Signals Function Selection Descriptions		
HCNTL 1	HCNTL 0	Description
0	0	Host can read or write the HPI control register, HPIC.
0	1	Host can read or write the address register, HPIA.
1	0	Host can read or write HPID. HPIA is postincremented by a word address (4 byte addresses).
1	1	Host can read or write HPID. HPIA is not affected.

Halfword Identification Select: HHWIL

[0142] Identifies first or second halfword of transfer (but not most significant or least significant --- this is specified by the HWOB bit in the HPIC register, described later in this chapter.) HHWIL is low for the first halfword and high for the second halfword.

Byte Enables: HBE-1:0

[0143] On HPID writes, the value of HBE-1:0 indicates which portions of the 32-bit word must be written. The value of HBE-1:0 is not important on HPIA or HPIC accesses or on HPID reads. On HPID writes, HBE0- enables the lsbyte in the halfword and HBE1-enables the MSByte in the halfword. Table 19 lists the valid combinations. For byte writes, only a one HBE- in either (but not both) of the halfword accesses may be enabled active-low. For halfword data writes, both the HBE-s must be held active-low in either (but not both) halfword accesses. For word accesses, both HBE- must be held active-low in both halfword accesses. No other combinations are valid. The selection of byte-enables and the endianness of the CPU (selected via the LENDIAN pin), determine the logical address implied by the access.

TABLE 19.

Byte Enables for HPI Data Write Access				
Data Write Type	HBE-1:0		Effective Address LSBs	Logical (binary)
	Second Write HHWIL=1	First Write HHWIL=0	Little Endian	Big Endian
Byte	11	10	00	11
Byte	11	01	01	10
Byte	10	11	10	01
Byte	01	11	11	00
Halfword	11	00	00	10
Halfword	00	11	10	00
Word	00	00	00	00

Read/Write Select: HR/W-

[0144] HR/W- is the host read write select input. Hosts must drive HR/W high to read HPI and low to write HPI. Hosts without either a read/write select output or a read or write strobe can use an address line for this function.

Ready: HRDY-

[0145] When active-low, HRDY- indicates that the HPI is ready for a transfer to be performed. When inactive-high HRDY- indicates that the HPI is busy completing the internal portion of current read access of a previous HPID write or HPID read prefetch access. HCS- enables HRDY; that is, HRDY is always low when HCS- is low.

Strobes: HCS-, HDS1-, HDS2-

[0146] CS-, HDS1-, HDS2- allow connection to hosts with either:

Single strobe output with read'write select.
Separate read and write strobe outputs. In this case, read or write select can be done by using different addresses for read and write select.

[0147] Figure 20 shows the equivalent circuit of the HCS-, HDS1 -, and H052-inputs. Used together, HCS-, HDS1, and HDS2- generate an active-low internal HSTRB- signal. Note that HSTRB- is only active low when both HCS- is active-low and either (but not both) HDS1 - and HDS2- are active-low. The falling edge of HSTRB- (when HAS- is tied inactive-high) samples HCNTL1:0, HHWIL, HR/W-, and HBE-1:0. Therefore, the latest of HDS1-, HDS2-, or HCS- that falls controls the sampling time. HCS- serves as the enable input for the HPI and must be low during an access. However, as the HSTRB- signal determines the actual boundaries between accesses, HCS- may stay low between successive accesses as long as both HDS1 - and HDS2- go inactive-high or go active-low.

[0148] Hosts with separate read and write strobes connect those strobes to either HDS1- or HDS2-. Hosts with a single data strobe connect it to either HDS1- or HDS2-, connecting the unused pin high. Regardless of HDS connections, HR/W- is still required to determine direction of transfer. Because HDS1-and HDS2- are internally exclusive-NORed, hosts with a high true data strobe can connect this to one of the HDS inputs with the other HDS-input connected low.

[0149] HSTRB- is used for three purposes:

1. On a read, its falling edge initiates HPI read accesses for all access types.
2. On a write, its rising edge initiates a HPI write accesses for all access types.
3. Falling edges latch HPI control inputs including: HHWIL, HR/W-, HBE-1:0, HCNTL1:0. HAS- also effects latching of control inputs;

See the description later herein.

[0150] Additionally, HCS- gates the HRDY output. In other words, a not-ready condition is indicated by the HRDY-pin driven inactive-high only if HCS- is active-low.

Address Strobe Input: HAS-

[0151] HAS- allows HCNTL1:0-, HBE-1:0, HR/W-, and HHWIL to be removed earlier in an access cycle, therefore allowing more time to switch bus states from address to data information. This feature facilitates interface to multiplexed address and data type buses. In this type of system, an ALE signal is often provided and would normally be the signal connected to HAS-.

[0152] Hosts with a multiplexed address and data bus connect HAS to their ALE pin or equivalent. HHWIL, HCNTLO/1, HBE-,and HR/W- are then latched on HAS-'s falling edge. When used, HAS- must precede the later of HCS-, HDS1-, or HDS2-. Hosts with separate address and data bus can connect HAS high. In this case, HHWIL, HCNTL0/1, and HR/W are latched by the later of HDS1-, HDS2-, or HCS-falling edge while HAS stays inactive (high).

Interrupt to Host: HINT-

[0153] Host interrupt output. Controlled by the HINT- bit in the HPIC. Driven high when the chip is being reset. This signal is described in more detail later herein.

HPI Bus Access

[0154] Figure 21 and Figure 22, show HPI access timing for the cases when HAS- is not and is used, respectively. HSTRB-represents the internally generated strobe described in Figure 20. HAD represents control signals typically driven by host address lines: HCNTL1:0, HR/W-, HHWIL, and HBE-1:0. HCNTL1:0 and HR/W should have the same

values for both halfword accesses. HHWIL is shown separately to indicate that it must be low for the first halfword transfer and high for the second. If HAS is not used (tied high as shown in Figure 21), the falling edge of HSTRB- latches these signals. If HAS is used as shown in Figure 22, the falling edge of HAS- latches these values. In this case, the falling edge of HAS- must precede the falling edge of HSTRB-. On a read, data is valid a delay after the falling edge of HSTRB-. If valid data is not already present in the HPID, the data is setup to the rising edge of HRDY- and held until the rising edge of HSTRB. On a write, the host must setup data with respect to the rising edge of HSTRB-.

HPI Registers

[0155]   Table 20 summarizes the three registers that the HPI utilizes for communication between the host device and the CPU. HPID contains the data that was read from the HPI memory if the current access is a read, or the data that will be written to HPI memory if the current access is a write. Contains the address in the HPI memory at which the current access occurs. As this address is a 30-bit word address the bottom two bits are unaffected by HPIA writes and are always read as 0.

TABLE 20.

HPI Register Description				
Register Acronym	Register Name	Host Read/Write	CPU Read/Write	CPU Read/Write (Hex Byte Address)
HPID	HPI Data	RW	-	-
HPIA	HPI Address	RW	-	-
HPIC	HPI Control	RW	RW	0188 0000

HPI Control Register (HPIC)

[0156]   Normally, the HPIC (See Figure 19A and Table 21) is normally the first register accessed to set configuration bits and initialize the interface. Thus, the HPIC is organized on the as a 32-bit register with the same high and low halfword contents. On a host write, both halfwords must be identical. Note that the low halfword and the high halfword are actually the same storage locations. No storage is allocated for the read only reserved values. Only CPU writes to the lower halfword affect HPIC values and HPI operation. Normally, the HPIC is normally the first register accessed to set configuration bits and initialize the interface. Thus, the HPIC is organized as a 32-bit register with the same high and low halfword contents.

TABLE 21.

HPI Control Register (HPIC) Bit Descriptions		
Bit	Description	Section
HWOB	Halfword Ordering Bit. If HWOB = 1, first halfword is least significant. If HWOB = 0, first halfword is most significant. HWOB affects both data and address transfers. Only the host can modify this bit. HWOB must be initialized before the first data or address register access.	
DSPINT	The host processor-to-CPU/DMA interrupt.	5.3.3. 1
HINT	DSP to Host Interrupt. Determines the state of the CPU HINT- output.	5.3.3. 2
HRDY	Ready signal to host. Not masked by HCS-like HRDY- pin. If HRDY=0 this indicates that the internal bus is waiting for a HPI data access request to complete.	5.3.2
FETCH	Host Fetch request. If the host writes a one to this bit, it requests a fetch into HPID the word at the address pointed to by HPIA. Always read as 0.	5.3.2

Software Handshaking Using HRDY and FETCH

[0157]   As described previously, the HRDY- pin can indicate to a host that a HPID access has not completed. For example, he current HPID access can be waiting for a previous HPID accesses write to complete or for a previous HPID prefetched read to complete. Also, the current HPID read access can be waiting for is requested data to arrive. However, the HRDY and FETCH bits in the HPIC allow for a software handshake that allows HPI connection in systems

where a hardware ready control is not desired. Any prefetched data remains until any HPI register is written or until after the next HPI read, whichever comes first. The FETCH and HRDY bits can be used to perform a read transfer as follows:

1. The host polls for the HRDY bit to be set.
2. The host writes the desired HPIA value. This step is skipped if HPIA is already set to the desired value.
3. The host writes a 1 to the FETCH bit.
4. The host polls for the HRDY bit to be set.
5. The host performs a HPID read operation. In this case, the HPI is already in the ready state (HRDY bit=1).

-   If this was a read with post increment, goto 4.
-   For a read from the same location, goto 3.
-   For a read to a different address, goto 2.

[0158] The HRDY bit alone can be used for write operations as follows:

1. The host polls for the HRDY bit to be set.
2. The host writes the desired HPIA value. This step is skipped if HPIA is already set to the desired value.
3. The host performs an HPID write operation.

-   For another write operation, goto 1.

DSPINT and HINT Function Operation

[0159] The host and the CPU can interrupt each other using bits in the HPIC register. This subsection presents more information about this process.

Host Device Using DSPINT to Interrupt the CPU

[0160] The host can interrupt the CPU by writing to the DSPINT bit in the HPIC. The DSPINT bit is tied directly to the internal DSPINT signal. By writing DSPINT=1 when DSPINT=0, the host causes a 0-to-1 transition on the DSPINT signal. If appropriately selected using the interrupt selector, this transition will be detected as an interrupt condition by the CPU. The CPU can clear the DSPINT bit by writing DSPINT=1. Neither a host nor a CPU HPIC write with DSPINT=0 effects either the DSPINT bit or signal.

CPU Using HINT- to Interrupt the Host

[0161] The CPU can send an active low interrupt condition on the HINT- signal by writing to the HINT bit in the HPIC. The HINT bit is inverted and tied directly to the HINT- pin. The CPU can set HINT- active low by writing HINT=1. The host can clear the HINT-to inactive-high bit by writing DSPINT=1. Neither a host nor a CPU HPIC write with HINT=0 effects either the HINT bit or HINT-signal. Note this bit is read twice on the host interface side, the first and second halfword reads by the host may yield different data if the CPU changes the state of one or both of these bits in between the two read operations.

Host Access Sequences

[0162] The host begins HPI accesses by first initializing HPIC, then HPIA, and then writing data to or reading data from HPID. Reading or writing HPID initiates an internal cycle that transfers the desired data between the HPID and the DMA Auxiliary Channel. Host access of any HPI register requires two halfword accesses on the HPI bus: the first with HHWIL low, and the second with HHWIL high. Typically, the host does not break the first halfword/second halfword (HHWIL low/high) sequence. If this sequence is broken improperly, data may be lost, and undesired operation may result. The first halfword access may have to wait for a previous HPI request to complete. Previous requests include HPID writes and pre-fetched HPID reads. Thus, the HPI will de-assert HRDY- high until the HPI can begin this request. The second halfword access will always have HRDY- active low since all previous accesses have completed by during the first halfword access.

Host Initialization of HPIC and HPIA (All the numeric values in the figures in this section are in Hexadecimal)

[0163]   Before accessing data, the host must first initialize HPIC, in particular the HWOB bit, and then HPIA (in this order, because HWOB affects the HPIA access). After initializing HWOB, the host can then write to HPIA with the correct halfword alignment. Table 22 and Table 23, illustrate the initialization sequence for HWOB=1 and HWOB=0, respectively. In this example, HPIA is set to 80001234h. Note that in all these accesses, HRDY bit in the HPIC is set (bit 19 and 3).

TABLE 22.

Initialization of HWOB=1 and HPIA								
Event	Value During Access					Value After Access		
	HD	HBE-1:0	HR/W-	HCNTL 1:0	HHWI L	HPID	HPIC	HPIA
Host writes HPIC 1st halfword	00 01	Xx	0	00	0	00090 009	????? ???	???? ????
Host writes HPIC 2nd halfword	00 01	Xx	0	00	1	00090 009	????? ???	???? ????
Host writes HPIA 1st halfword	12 34	Xx	0	01	0	00090 009	????1 234	???? ????
Host writes HPIA 2nd halfword	80 00	Xx	0	01	1	00090 009	80001 234	???? ????

TABLE 23.

Initialization of HWOB=0 and HPIA								
Event	Value During Access					Value After Access		
	HD	HBE-1:0	HR/W-	HCNTL 1:0	HHWI L	HPIA	HPIC	HPID
Host writes HPIC 1st halfword	0000	Xx	0	00	0	00080 008	????? ???	???? ????
Host writes HPIC 2nd halfword	0000	Xx	0	00	1	00080 008	????? ???	???? ????
Host writes HPIA 1st halfword	8000	Xx	0	01	0	00080 008	8000? ???	???? ????
Host writes HPIA 2nd halfword	1234	Xx	0	01	1	00080 008	80001 234	???? ????

HPID Read Access without Auto-Increment

[0164]   Assume that once the HPI is initialized as described earlier herein, that the host wishes to do a read access to that address without an autoincrement. Assume that the host wants to read the word at address 80001234h and that the word value at that location is 789ABCDEh. Table 24 and Table 25, illustrate this access for HWOB=1 and HWOB=0, respectively. On the first halfword accesses, the HPI waits for any previous requests to complete. During this time, HRDY- is held inactive-high. Then, the HPI sends the read request to the DMA Auxiliary Channel. If no previous requests are pending, this occurs with the falling edge of HSTRB-. HRDY- remains high until the DMA Auxiliary Channel loads the requested data into HPID. Because all DMA Auxiliary Channel reads are word reads, at the beginning of the second read access, the data is already present in HPID. Thus, the second halfword HPID read will never encounter a not ready condition and HRDY- will remain low. Note that the byte enables are not important as the HPI only does word reads.

TABLE 24.

Read Access to HPI Without Auto-Increment: HWOB=1								
Event	Value During Access						Value After Access		
	HD	HBE-1:0	HR /W-	HCNTL1:0	HRD Y-	HHWI L	HPIC	HPIA	HPID
Host reads 1st halfword Data Not Ready	????	xx	1	11	1	0	00010001	80001234	???? ????
Host read 2nd halfword Data Ready	BCDE	xx	1	11	0	0	00090009	80001234	789A BCDE
Host reads 2nd halfword	789A	xx	1	11	0	1	00090009	80001234	789A BCDE

TABLE 25.

Read Access to HPI Without Auto-Increment: HWOB=0								
Event	Value During Access						Value After Access		
	HD	HBE-1:0	HR/W-	HCNTL1:0	HRDY-	HHWIL	HPIC	HPIA	HPI D
Hos t reads 1st halfword Data Not Ready	????	xx	1	11	1	0	00000000	80001234	??? ??? ??
Host read 2nd halfword Data Ready	789A	xx	1	11	0	0	00080008	80001234	789 ABCDE
Host reads 2nd halfword	BCDE	xx	1	11	0	1	00080008	80001234	789 ABCDE

HPID Read Access with Auto-Increment

[0165]   The autoincrement feature results in efficient sequential host accesses. For both HPID read and write accesses, this removes the need for the host to load incremented address into HPIA. For read accesses, the data pointed to by the next address begins being fetched immediately after the completion of the current read. Because the intervals between successive reads is used to prefetch data, the latency for the next access is reduced. Any prefetched data remains until any HPI register is written or until after the next HPI read, whichever comes first. Prefetching can also occur after a host write of FETCH=1 to the HPIC. If the next HPI access is a HPID read, then the data is not re-fetched

and the pre-fetched data is sent to the host. Otherwise, the HPI must still wait for the prefetch to complete.

[0166] Table 26 shows a read access with auto-increment. After the first halfword access is complete (with the rising edge of the first HSTRB-), the address increments to the next word or 80001238h. Assume that the data at that location is 87654321h. That data is pre-fetched and loaded into HPID. Prefetching begins on the rising edge of HSTRB- on the second halfword read.

TABLE 26.

Read Access to HPI With Auto-Increment: HWOB=1								
Event	Value During Access						Value After Access		
	HD	HB E-1: 0	HR /W-	HC NT L1:0	HR DY-	H H W I L	HPIC	HPIA	HPID
Host reads 1st halfword Data Not Ready	????	Xx	1	10	1	0	00010001	80001234	????????
Host read 2nd halfword Data Ready	BCDE	Xx	1	10	0	0	00090009	80001234	789ABCDE
Host reads 2nd halfword	789A	Xx	1	10	0	1	00090009	80001238	789ABCDE
Pre-fetch Data Not Read	????	Xx	x	xx	0	x	00010001	80001238	789ABCDE
Pre-fetch Data Ready	????	Xx	x	xx	0	x	00090009	8000123 8	87654321

TABLE 27.

Read Access to HPI With Auto-Increment: HWOB=0								
Event	Value During Access						Value After Access		
	HD	HBE-1:0	HR/W-	HCNTL1:0	HRDY-	HHWIL	HPIC	HPIA	HPID
Host reads 1st halfword Data Not Ready	????	Xx	1	10	1	0	00000000	80001234	????????
Host read 2nd halfword Data Ready	789A	Xx	1	10	0	0	00080008	80001234	789ABCDE

TABLE 27. (continued)

Read Access to HPI With Auto-Increment: HWOB=0								
Event	Value During Access						Value After Access		
	HD	HBE-1:0	HR/W-	HCNTL1:0	HRDY-	HHWIL	HPIC	HPIA	HPID
Host reads 2nd halfword	BCDE	Xx	1	10	0	1	00080008	80001238	789ABCDE
Pre-fetch Data Not Read	????	Xx	x	xx	0	x	00000000	80001238	789ABCDE
Pre-fetch Data Ready	????	X x	x	xx	0	x	00080008	80001238	87654321

Host Data Write Access without Auto Increment

[0167]   During a write access to the HPI, the first halfword portion of HPID (LSHalfword or MSHalfword as selected by HWOB) is overwritten by the data coming from the host and the first HBE-1:0 latched while the HHWIL pin is low. The second halfword portion of HPID is overwritten by the data coming from the host and the second HBE-1:0 pair is latched while the HHWIL pin is high. At the end of this write access (with the second rising edge HSTRB-), HPID is transferred as a 32-bit word to the address specified by HPIA with the four related byte-enables.

[0168]   Table 28 and Table 29 illustrate an HPID write access with HWOB=1 and HWOB=0, respectively. The host writes 5566h to the 16 LSBs of location 80001234h, which is already pointed to by HPIA. This location is assumed to start with the value 0. The HPI holds of the host until any previous transfers complete by setting HRDY- inactive-high. Also, once the first halfword becomes ready, the second halfword does not encounter any not ready time. If there are no pending writes waiting in HPID, then write accesses normally proceed without a not ready time. Note that the HBE-1:0 is only enabled for the transfer which transfers the 16 LSBs.

TABLE 28.

Write Access to HPI Without Auto-Increment: HWOB=1										
Event	Value During Access						Value After Access			
	HD	HBE-1:0	HRW-	HCNTL1:0	HRDY-	HHWIL	HPIC	HPIA	HPID	Location 80001234
Host writes HPID 1st halfword Waiting for Previous	5566	00	0	11	1	0	00010001	80001234	????5566	00000000
Host wrotes HPID 1st halfword Ready	5566	00	0	11	0	0	00090009	80001234	????5566	00000000
Host writes HPIA 2nd halfword	wxyz	11	0	11	0	1	00090009	80001234	wxyz5566	00005566

TABLE 29. Write Access to HPI Without Auto-Increment: HWOB=0

Event	Value During Access									Value After Access			
	HD	HBE-1:0	HR/W-	HCNTL1:0	HRDY-				HHWIL	HHPIC	HHPIA	HHPID	Location 80001234
Host writes HPID 1st half word Waiting for Previous	wxyz	11	0	11	1	0	00000000	80001234	wxyz ????				00000000
Host wrote HPID 1st half word Ready	wxyz	11	0	11	0				0	0 0 0 8 0 0 0 8	8 0 0 0 1 2 3 4	w x y z ? ? ? ?	000 000 00
Host writes HPIA	5566	00	0	11	0				1	0 0 0 8	8 0 0 0	w x y z	000 055 66

40

2nd half word							0	1	5	
							0	2	5	
							0	3	6	
							8	8	6	
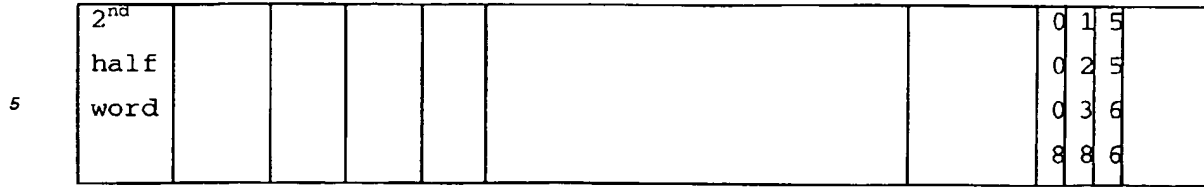
HPID Write Access with Auto-Increment

[0169]   Table 30 and Table 31 illustrate a host write data with auto-increment for HWOB=1 and HWOB=0, respectively. These examples are identical to the ones in 5.4.4, with the exception of the HCNTL1:0 value and a subsequent write of 33 to the most significant byte of the word at address 80001238. The increment happens on the rising edge of HSTRB- on the next HPID write access. If the next access is an HPID or HPIC access or a HPID read the autoincrement does not occur.

TABLE 30.

Write Access to HPI With Auto-Increment: HWOB=1											
Event	Value During Access						Value After Access				
	HD	HBE-1:0	HR/W-	HCNTL1:0	HRDY-	HHWIL	HPIC	HPIA	HPID	Location 80001234	Location 80001238
Host writes HPID 1st halfword Waiting for Previous	5566	00	0	10	1	0	00010001	80001234	????5566	00000000	00000000
Host wrotes HPID 1st halfword Ready	5566	00	0	10	0	0	00090009	80001234	????5566	00000000	00000000
Host writes HPIA 2nd halfword	wxyz	11	0	10	0	1	00090009	80001234	wxyz5566	00005566	00000000
Host writes HPID 1st halfword Waiting for Previous	nopq	11	0	10	1	0	00010001	80001234	wxyznopq	00005566	00000000
Host writes HPID 1st halfword Ready	nopq	11	0	10	0	1	00090009	80001238	wxyznopq	00005566	00000000
Host writes HPIA 2nd halfword	33rs	01	0	10	0	1	00090009	80001238	33rsnopq	00005566	33000000

TABLE 31.

Write Access to HPI Without Auto-Increment WHOB=0											
Event	Value During Access						Value After Access				
	HD	HBE-1:0	HR/W-	HCNTL1:0	HRDY-	HHWIL	HPIC	HPIA	HPID	Location 80001234	Location 80001238
Host writes HPID 1st halfword Waiting for Previous	5566	00	0	1 0	1	0	00000000	80001234	????5566	00000000	00000000
Host wrotes HPID 1st halfword Ready	5566	00	0	1 0	0	0	00080008	80001234	????5566	00000000	00000000
Host writes HPIA 2nd halfword	wxyz	11	0	1 0	0	1	00080008	80001234	wxyz5566	00005566	00000000
halfword											
Host writes HPID 1st halfword Waiting for Previous	33rs	11	0	1 0	1	0	00000000	80 00 12 34	33rs5566	00005566	00000000
Host writes HPID 1st halfword Ready	33rs	11	0	1 0	0	1	00080008	80 00 12 38	33rs5566	00005566	00000000
Host writes HPIA 2nd halfword	nopq	01	0	1 0	0	1	00080008	80 00 12 38	33rsnopq	00005566	33000000

Single Halfword Cycles

[0170]   In normal operation, every transfer must consist of two halfword accesses. However, to speed operation the user may perform single-halfword accesses. These can be useful in several cases:

HPIC: Note that in Table 22 that the entire HPIC was written correctly after the first write. When writing the HPIC, the host does not have to be concerned about HHWIL, nor does it have to perform two consecutive writes to both halfwords. Similarly, the host can choose to only read the HPIC once since both halves contain the same value.
HPIA: Note that Table 22, that the portion of HPIA accesses as selected by HHWIL and HWOB automatically is updated after each halfword access. Thus, to change the only the upper or lower 16 bits of HPIA the host only needs select which half it wants to modify through a combination of HHWIL and HWOB. Similarly, the host can choose to only read the desired half of HPIA.
HPID Read Accesses: Read accesses are actually triggered by the first halfword access (HHWIL is low). Thus, if on reads the host is only interested in the first halfword (least or most significant selected by HWOB), it does not need to request the second address. However, pre-fetching will not occur unless the second halfword is also read. A subsequent read of the first halfword (HHWIL low) or the write of a new value to HPIA will override any previous prefetch request. On the other hand, a read of only the second halfword (HHWIL high) is not allowed and can result in undefined operation.

[0171]   Write Accesses: Write accesses are triggered by the second halfword access (HHWIL is high). Thus, if the host only desired to change the portion of HPID selected by HHWIL high (and the associated byte enables) during consecutive write accesses, only a single cycle would need to be initiated. This techniques primary use would be for memory fills. To do this the host would write both halfwords of the first write access with HBE1:0=00. On subsequent write accesses, the host would make sure it writes the same value to the portion of HPID selected by HHWIL as did the first write access. In this case, the host would perform auto-incrementing writes (HCNTL1:0=01) on all write accesses.

Access of HPI Memory During Reset

[0172]   The HPI cannot be used while the chip is in reset. However, certain boot modes may allow the host to write the CPU's memory space (including configuring EMIF configuration registers to configure external memory before accessing it). Although the device is not in reset, the CPU itself is in reset until the boot completes. Refer to the bootload section later herein for full details.

EMIF

Overview

[0173]   The External Memory Interface (EMIF) supports a glueless interface to several external devices including:

Synchronous Burst SRAM (SBSRAM) running at 1x and ½x the CPU clock rate.
Synchronous DRAM (SDRAM) running at ½x the CPU clock rate.
Asynchronous devices including asynchronous SRAM, ROM, and FIFOs. The EMIF provides highly programmable timing to these interfaces.

[0174]   The External Memory Interface (EMIF) services requests of the external bus from four requesters as shown in Figure 4:

The on-chip Program memory controller (PMC) that services CPU program fetches.
The on-chip Data memory controller (DMC) that services CPU data fetches.
The on-chip DMA Controller.
An external shared-memory device.

[0175]   If multiple requests arrive simultaneously, the EMIF must prioritizes them and perform the necessary cycles. A block diagram of the EMIF is shown in Figure 23. Table 32 describes the EMIF signals.

TABLE 32.

EMIF Signal Descriptions		
Pin	(I/O/Z)	Description
CLKOU T1	O	Clock. Clock output - the CPU clock rate.
CLKOU T2	O	Clock. Clock output - ½ the CPU clock rate.
ED(31 :0)	I/O /Z	Data I/O. 32-bit data input/output from external memories and peripherals.
EA(21 :2)	O/Z	External Address output. Drives bits 21-2 of the byte address.
/CEO	O/Z	External /CEO Chip Select. Active low chip select for CE space 0.
/CE1	O/Z	External /CE1 Chip Select. Active low chip select for CE space 1.
/CE2	O/Z	External /CE2 Chip Select. Active low chip select for CE space 2.
/CE3	O/Z	External /CE3 Chip Select. Active low chip select for CE space 3.
/BE (3 :0)	O/Z	Byte Enables. Active low byte strobes. Individual bytes and halfwords can be selected for both read and write cycles. Decoded from 2 LSBs of the byte address.
/ARDY	I	Ready. Active low asynchronous ready input used to insert wait states for slow memories and peripherals.
/AOE	O/Z	Output Enable. Active low output enable for asynchronous memory interface.
/AWE	O/Z	Write Strobe. Active low write strobe for asynchronous memory interface.
/ARE	O/Z	Read Strobe. Active low read strobe for asynchronous memory interface.
/SSAD S	O/Z	Address Strobe. Active low address strobe/enable for SBSRAM interface.
/SSOE	O/Z	Output Enable. Output buffer enable for SBSRAM interface.
/SSWE	O/Z	Write Enable. Active low write enable for SBSRAM interface.
SSCLK	O/Z	Clock. SBSRAM interface clock; equivalent to CLKOUT1 or CLKOUT2 as selected by the user.
/SDRA S	O/Z	Row Address Strobe. Active low /RAS for SDRAM memory interface.
/SDCA S	O/Z	Column Address Strobe. Active low /CAS for SDRAM memory interface.
/SDWE	O/Z	Write Enable. Active low /W for SDRAM memory interface.
SDA10	O/Z	SDRAM A10 Address Line. Address line/auto-precharge disable for SDRAM memory.
SDCLK	O/Z	Clock. SDRAM interface clock- ½ the CPU clock rate. Equivalent to CLKOUT2.
/HOLD	I	Hold. Active low external bus hold (tristate) request.
/HOLD A	O	Hold Acknowledge. Active low external bus hold acknowledge.

Resetting the EMIF

[0176] A hardware reset using the /RESET pin on the device forces all register values to their reset state. During reset, all outputs are driven to their inactive levels, with the exception of the clock outputs (SDCLK, SSCLK, CLKOUT1 and CLKOUT2), which continue to be driven such that external logic which is using these signals can be synchronized.

EMIF Registers

[0177] Control of the EMIF and the memory interfaces it supports is maintained through a set of memory mapped registers within the EMIF. A write to any EMIF register will not complete until all pending EMIF accesses which use that register have completed. The memory mapped registers are shown in Table 33.

TABLE 33.

EMIF Memory Mapped Registers	
Byte Address	Name
0x01800000	EMIF Global Control
0x01800004	EMIF CE1 Space Control
0x01800008	EMIF CE0 Space Control
0x0180000C	Reserved
0x01800010	EMIF CE2 Space Control
0x01800014	EMIF CE3 Space Control
0x01800018	EMIF SDRAM Control
0x0180001C	EMIF SDRAM Refresh Period

EMIF Global Control Register

[0178] The EMIF Global Control Register (Figure 24 and Table 34) configures parameters common to all the CE spaces.

TABLE 34.

EMIF Global Control Register Bit Field Description	
Field	Description
SDCINV	SDCLK polarity SDCINV = 0, SDCLK output is inverted from internal SDCLK SDCINV = 1, SDCLK output is identical to internal SDCLK
CLK2INV	CLKOUT2 polarity CLK2INV = 0, CLKOUT2 output is inverted from internal CLKOUT2 CLK2INV = 1, CLKOUT2 output is identical to internal CLKOUT2
/ARDY	Value of /ARDY input
/HOLD	Value of /HOLD input
/HOLDA	Value of /HOLDA output
NOHOLD	External HOLD disable (1 = hold disabled; 0 = hold enabled)
SDCEN	SDCLK enable SDCEN =0, SDCLK held high SDCEN =1, SDCLK enabled to clock
SSCEN	SSCLK enable SSCEN=0, SSCLK held high SSCEN=1, SSCLK enabled to clock
CLK1EN	CLKOUT1 enable CLK1EN=0, CLKOUT1 held high CLK1EN=1, CLKOUT1 enabled to clock
CLK2EN	CLKOUT2 enable CLK2EN=0, CLKOUT2 held high CLK2EN=1, CLKOUT2 enabled to clock
SSCRT[1]	SBSRAM clock rate select SSCRT=0, SSCLK ½x CPU clock rate SSCRT=1, SSCLK 1x CPU clock rate

[1] The reset value of this field may be changed for the boot configuration of the particular part. See Chapter 7 Boot Configuration, Reset, and Memory Map for the available boot configurations. This feature allows various power-up configurations of memory.

TABLE 34. (continued)

EMIF Global Control Register Bit Field Description	
Field	Description
RBTR8	Requester arbitration mode RBTR8=0, requester controls EMIF until a high priority request occurs RBTR8=1, requester controls EMIF for a minimum of eight accesses
MAP	Map mode, contains the value of the memory map mode of the device

6.2.2 CE Space Control Registers

[0179] The four CE Space Control Registers (Figure 25 and Table 35) correspond to the four CE spaces supported by the EMIF. The MTYPE field identifies the memory type for the corresponding CE space. If MTYPE selects SDRAM or SBSRAM, the remaining fields in the register do not apply. If an asynchronous type is selected (ROM or Asynchronous), the remaining fields specify the shaping of the address and control signals for access to that space. Modification of a CE Space Control Register does not occur until that CE space is inactive.

TABLE 35.

EMIF CE(0/1/2/3) Space Control Registers Bitfield Description	
Field	Description
READ SETUP WRITE SETUP	Setup width. Number of CLKOUT1 cycles of setup for address (EA) and byte enables (/BE(0-3)) before read strobe (/ASRE) or write strobe (/ASWE) falling. On the first access to a CE space this is also the setup after /CE falling.
READ STROBE WRITE STROBE	Strobe width. The width of read strobe (/ASRE) and write strobe (/ASWE) in CLKOUT1 cycles.
READ HOLD RITE HOLD	Hold width. Number of CLKOUT1 cycles that address (EA) and byte strobes (/BE(0-3)) are held after read strobe (/ASRE) or write strobe (/ASWE) rising.
TA	Turn-around width. The number of CLKOUT1 cycles between reads and writes or between accesses to different CE spaces. Measured from when /OE rises. One turn-around cycle is inserted between a write followed by a read. One turn-around cycle is inserted between consecutive accesses to two different CE spaces from the same requester. TA=00b, 1 turn-around cycle. TA=01b, 2 turn-around cycles. TA=10b, 3 turn-around cycles. TA=11b, 4 turn-around cycles.
MTYPE[2]	Memory Type. MTYPE=000b, 8-bit wide ROM [3] MTYPE=001b, 16-bit wide ROM [3] MTYPE=010b, 32-bit wide Asynchronous Interface MTYPE=011b, 32-bit wide SDRAM [4] MTYPE=100b, 32-bit wide SBSRAM MTYPE=other, reserved

[2] The reset value of this field may be changed for the boot configuration of the particular part. See Chapter 7 Boot Configurations, Reset, and Memory Map for the available boot configurations. This feature allows various power-up configurations of memory.
[3] Only available in CE1 Space. Reserved in CE0/2/3 Space Control Registers.
[4] Only available in CE0, CE2, and CE3. Reserved in CE1.

SDRAM Control Register

[0180] The SDRAM Control Register controls SDRAM parameters for all CE spaces which specify an SDRAM mem-

ory type in the MTYPE field of its associated CE Space Control Register (Figure 25). Since the SDRAM Control Register controls all SDRAM spaces, each space must contain SDRAM with the same refresh and page characteristics.

TABLE 36.

EMIF SDRAM Control Register Bitfield Description	
Field	Description
TRC TRP TRCD	Specifies t RC value of the SDRAM in CLKOUT2 cycles. TRC = t RC - 1. Specifies t RP value of the SDRAM in CLKOUT2 cycles. TRP = t RP - 1. Specifies t RCD value of the SDRAM in CLKOUT2 cycles. TRCD = t RCD - 1.
INIT	Forces an initialization of all SDRAM present. See section 6.3.1. INIT = 0, no effect. NIT = 1, initialize SDRAM in each CE space configured for SDRAM.
RFEN	Refresh Enable. RFEN = 0, SDRAM refresh disabled. RFEN = 1, SDRAM refresh enabled.
SDWID	SDRAM Width Select SDWID=0, Each External SDRAM Space Consists of 4 8-bit SDRAMs SDWID=1, Each External SDRAM Space Consists of 2 16-bit SDRAMs

SDRAM Timing Register

[0181] The SDRAM refresh period register (Figure 27 and Table 37) controls the refresh PERIOD for SDRAM in terms of CLKOUT2 cycles (½x the CPU clock rate). Optionally, the refresh period can send an interrupt to the CPU. Thus, this counter may be used as a general purpose timer if SDRAM is not used by the system. The COUNTER value can be read by the CPU. When the counter reaches zero, it is automatically reloaded with the PERIOD, and an interrupt is sent to the CPU.

TABLE 37.

EMIF SDRAM Timing Register Bitfield Description	
Field	Description
PERIOD	Refresh period in CLKOUT2 cycles.
COUNTER	Current value of the refresh counter.

SDRAM Interface

[0182] The EMIF supports 16 Mbit, 2 bank and 64Mbit, 4 bank SDRAM offering system designers an interface to high speed and high density memory. The following sections describe the EMIF SDRAM interface. The EMIF supports the SDRAM commands shown in Table 38. 16M bit and 64 Mbit SDRAM interfaces are shown in Figure 28 and Figure 29.
[0183] Table 40 describes the pin connection and related signals specific to SDRAM operation. Table 39 shows all of the possible SDRAM configurations available via the EMIF.

TABLE 38.

EMIF SDRAM Commands	
Command	Function
DCAB	Deactivate (also known as pre-charge) all banks
ACTV	Activate the selected bank and select the row
READ	Input the starting column address and begin the read operation
WRT	Input the starting column address and begin the write operation
MRS	Mode Register Set, configures SDRAM mode register

TABLE 38. (continued)

EMIF SDRAM Commands	
Command	Function
REFR	Auto refresh cycle with internal address

In Figure 28 and Figure 29, CE0, CE2, and CE3 are useable for SDRAM. Thus, n=0, n=2, or n=3. Also, note that CLKOUT2 has the same timing as SDCLK and may used in systems where multiple CE-spaces of SDRAM are used and additional drive capability is necessary.

TABLE 39.

SDRAM Memory Population				
SDRAM size	SDRAM banks	SDRAM width	Devices per CE space	Memory Size per CE space
16 Mbit	2	16 bit	2	4 Mbytes
16 Mbit	2	8 bit	4 8 Mbytes	
64 Mbit	4	16 bit	2	16 Mbytes

TABLE 40.

SDRAM Control Pins		
EMIF Signal	SDRAM Signal	SDRAM Function
SDA10	A10	Address line A10/auto-precharge disable. Serves as a row address bit during ACTV commands and also disables the auto-pre-charging function of SDRAM.
/SDRAS	/RAS	Row Address Strobe and Command Input. Latched by the rising CLK to determine current operation. Only valid if /CS is active- low during that clock edge.
/SDCAS	/CAS	Column Address Strobe and Command Input. Latched by the rising CLK to determine current operation. Only valid if /CS is active- low during that clock edge.
/SDWE	/WE	W rite Strobe and Command Input. Latched by the rising CLK to determine current operation. Only valid if /CS is active- low during that clock edge.
/BE[3:0]	DQM[3:0]	Data / output mask. DQM is an input/output buffer control signal. It disables writes and tri-states outputs during reads when high. DQM has a 2 CLK latency on reads and a 0 CLK latency on writes. DQM pins serve essentially as byte strobes and are connected to /BE[3:0] outputs.
/CE3 or /CE2 or /CE0	/CS	Chip select and command enable. /CS must be active-low for a command to be clocked into the SDRAM. /CS does not affect data input or output once a write or read has begun.
-	CKE	CKE clock enable. Tied active high when interface to EMIF to always enable clocking.
CLKOUT2	CLK	SDRAM clock input. Runs at ½ the CPU clock rate.
SDCLK	CLK	SDRAM clock input. Runs at ½ the CPU clock rate. Equivalent to CLKOUT2.

SDRAM Initialization

[0184]   The EMIF performs the necessary functions to initialize SDRAM if any of the CE spaces are configured for SDRAM. An SDRAM initialization is requested by a write of 1 to the INIT in the EMIF SDRAM Control Register. The

actual sequence of events of an initialization is as follows:

1. Perform DCAB command to all CE spaces configured as SDRAM.
2. Perform 3 refresh commands.
3. Perform MRS command to all CE spaces configured as SDRAM.

[0185] The DCAB cycle is performed immediately after reset, provided the /HOLD input is not active (a host request). If /HOLD is active, the DCAB command will not be performed until the hold condition is removed. The external requester should not attempt to access any SDRAM banks in this case, unless it performs SDRAM initialization and control itself.

Monitoring Page Boundaries

[0186] Because SDRAM is a paged memory type, the EMIF SDRAM controller monitors the active row of SDRAM so that row boundaries are not crossed during the course of an access. To accomplish this monitoring, the EMIF stores the address of the open page, and performs compares against that address for subsequent accesses to the SDRAM bank. This storage and comparison is performed independently for each CE space.

[0187] The number of address bits compared is a function of the page size programmed in the SDWID field in the EMIF SDRAM Control Register. If SDWID=0 in the SDRAM Control Register, the EMIF expects CE Spaces configured as SDRAM to have 4 8-bit wide SDRAMs that have page sizes of 512. Thus, the logical byte address bits compared are 26:11. If SDWID=1, the EMIF expects CE Spaces with SDRAM to have 2 16-bit wide SDRAMs that have page sizes of 256. Thus, the logical byte address bits compared are 23:10.

[0188] If, during the course of an access, a page boundary is crossed, the EMIF performs a DCAB command and starts a new row access. Also, a change in direction of an access (read to write or write to read) causes a page miss. Note that simply ending the current access is not a condition which forces the active SDRAM row to be closed. The EMIF speculatively leaves the active row open until it becomes necessary to close it. This feature decreases the de-activate-reactivate overhead and allows the interface to fully capitalize on address locality of memory accesses.

Refresh

[0189] The RFEN bit in the SDRAM Control Register (Table 36) selects the SDRAM refresh mode of the EMIF. A value of 0 in the RFEN field disables all EMIF refreshes, the user must insure that refreshes are implemented in an external device. A value of 1 in the RFEN field enables the EMIF to perform refreshes of SDRAM as described below.

[0190] Refresh commands (REFR) enable all /CE signals for all CE spaces selected to use SDRAM with the MTYPE field of the CE Space Control Register (Figure 25). REFR is automatically preceded by a DCAB command. This ensures all CE spaces selected with SDRAM are deactivated. Following the DCAB command, the EMIF begins performing "trickle" refreshes, at a rate defined by the PERIOD value in the EMIF SDRAM Control register provided no other SDRAM access is pending.

[0191] The SDRAM interface monitors the number of refresh requests posted to it and performs them. Within the EMIF SDRAM control block, a small 2 bit counter monitors the backlog of refresh requests. The counter increments once for each refresh request and decrements once for each refresh cycle performed. The counter will saturates at the value of 11, and also at 00. At reset, the counter is automatically set to 11, to ensure that several refreshes occur before accesses begin.

[0192] As depicted in Figure 30, the EMIF SDRAM controller prioritizes SDRAM refresh requests with other data access requests posted to it from the EMIF requesters. The following rules are followed:

A counter value of 11 invalidates the page information register, forcing the controller to close the current SDRAM page. The value of 11 indicates an urgent refresh condition. Thus, the EMIF SDRAM controller performs three REFR commands thereby decrementing the counter to 00 following the DCAB command before proceeding with the remainder of the current access. If SDRAM is present in multiple CE spaces, the DCAB-refresh sequence occurs in all spaces containing SDRAM.
During idle times on the SDRAM interface(s), if no request is pending from the EMIF, the SDRAM interface performs REFR commands as long as the counter value is nonzero. This feature reduces the likelihood of having to perform urgent refreshes during actual SDRAM accesses later. Note that if SDRAM is present in multiple CE spaces, this refresh occurs only if all interfaces are idle with invalid page information.

[0193] The EMIF SDRAM interface performs CAS-before-RAS refresh cycles for SDRAM. Some SDRAM manufacturers call this auto-refresh. Prior to a REFR command, a DCAB command is performed to all CE spaces specifying SDRAM to ensure that all active banks are closed. Page information is always invalid before and after a REFR com-

mand; thus a refresh cycle always forces a page miss. Note that a deactivate cycle is required prior to the refresh command.

Mode Register Set

[0194] As depicted in Figure 31, the EMIF automatically performs a DCAB command followed by a MRS command whenever the INIT field in the EMIF SDRAM Control Register is set. INIT can be set by device reset, or by a user write. Like DCAB and REFR commands, MRS commands are performed to all CE spaces configured as SDRAM through the MTYPE field to hold. Following the MRS cycle, the INIT bit clears itself to prevent multiple MRS cycles. Following an should return it to its original value before returning control of the bus to the EMIF. Alternatively, the user could poll the HOLD and HOLDA bits in the EMIF Global control register and upon detecting completion of an external hold re-initialize the EMIF by setting the INIT bit in the EMIF SDRAM Control Register.

[0195] The EMIF always uses a Mode Register value of 0x0030 during a MRS command. Table 41 shows the mapping between mode register bits, EMIF pins, and the mode register value.

[0196] Table 42 shows the SDRAM configuration selected by this mode register value.

TABLE 41.

Mode Register Value

Mode Register Bit	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMIF Pins	EA15	EA 14	EA13	SDA10	EA11	EA10	EA9	EA8	EA7	EA6	EA5	EA4	EA3	EA2
Field	Reserved				Write Burst Length	Reserved		Read Latency			S/I	Burst Length		
Value	0	0	0	0	0	0	0	0	1	1	0	0	0	0

TABLE 42.

Implied SDRAM Configuration by MRS Value	
Field	Selection
Write Burst Length	1
Read Latency	3
Serial/Interlea ve Burst Type	Serial
Burst Length	1

Address Shift

[0197] Because the same EMIF pins address the row and column address, the EMIF interface appropriately shifts the address in row and column address selection. Table 43 shows the translation between bits of the byte address and how they appear on the EA pins for row and column addresses. SDRAMs use the address inputs for control as well as address. W ith this consideration, the following items clarify the figure:

The address line that corresponds to the SDRAM's Bank Select bit (EA11 on 16Mbit SDRAM; EA13, EA12 on 64Mbit SDRAM) is latched internally by the SDRAM controller. This ensures that the bank select remains correct during READ and WRT commands. Thus, EMIF maintains these values as shown in both row and column addresses.
EMIF forces the address bit below bank select (SDA10) to be low unless /RAS is active low, yet high during DCAB commands at the end of a page of accesses. This prevents the auto precharge from occurring following a READ or WRT command.

TABLE 43. Byte Address to EA Mapping for SDRAM RAS and CAS

(The RAS and CAS values indicate the bit of the byte address present on the corresponding EA pin during a RAS or CAS cycle.)

SDRAM width	SDWID	DRAM command	EA2 - EA17	EA16	EA15	EA14	EA13	SDA10	EA11	EA10	EA9	EA8	EA7	EA6	EA5	EA4	EA3	EA2
x16	1	RAS	■	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
		CAS	■	24	23	22	21	20	19	18	9	8	7	6	5	4	3	2
x8	0	RAS	■		24	23	22	21	20	19	18	17	16	15	14	13	12	11
		CAS	■	24	23	22	21	20	10	9	8	7	6	5	4	3	2	

Key:

■ Bit is internally latched during ACTV command

– – – Reserved for future use.

Undefined.

Timing Requirements

[0198] Five SDRAM timing parameters decouple the EMIF from SDRAM speed limitations. Three of these parameters are programmable via the EMIF SDRAM control register; the remaining two are assumed to be static values as shown in Table 44. The three programmable values assure that EMIF control of SDRAM obeys these minimum timing requirements. Consult the SDRAM datasheet for the parameters appropriate for the particular SDRAM.

TABLE 44.

SDRAM Timing Parameters		
Parame ter	Description	Value in CLKOUT2 Cycles
t_{RC}	REFR command to ACTV, MRS, or subsequent REFR command	TRC + 1

EP 0 901 081 A2

TABLE 44. (continued)

SDRAM Timing Parameters		
Parameter	Description	Value in CLKOUT2 Cycles
t_{RCD}	ACTV command to READ or WRT command	TRCD + 1
t_{RP}	DCAB command to ACTV, MRS, or REFR command	TRP +1
t_{RAS}	ACTV command to DEAC to DCAB command	7
t_{nEP}	Overlap between read data and a DCAB command	2

Deactivation

[0199]   As depicted in Figure 32, the SDRAM deactivation (DCAB) is performed after a hardware reset, or when INIT=1 in the EMIF SDRAM Control Register. This cycle is also required by the SDRAMs prior to REFR, MRS, and when a page boundary is crossed. During the DCAB command, SDA10 is driven high to ensure that all SDRAM banks are deactivated.

SDRAM Read

[0200]   As depicted in Figure 33, during a SDRAM read the selected bank is activated with the row address during the ACTV command. In this example, four read commands are performed to four different column addresses. The EMIF uses a CAS latency of 3 and a burst length of 1. The 3 cycle latency causes data to appear 3 cycles after the corresponding column address. Following the last column access, a DCAB cycle is performed to deactivate the bank. An idle cycle is inserted between the final read command and the DCAB command to meet SDRAM timing requirements. Note that the transfer of data completes during and past the DCAB command. If no new access is pending, the DCAB command is not performed until such time that the page information becomes invalid. The values on EA[13:11] during column accesses and the DCAB command are the values latched during the ACTV command.

SDRAM Write

[0201]   As depicted in Figure 34, all SDRAM writes are burst length of 1. The bank is activated with the row address during the ACTV command. There is no latency on writes so data is output on the same cycle as the column address. Writes to invalid bytes are disabled via the appropriate DQM inputs; this feature allows for byte and halfword writes. Following the final write command an idle cycle is inserted to meet SDRAM timing requirements. The bank is then deactivated with a DCAB command and the memory interface can begin a new page access. If no new access is pending, the DCAB command is not performed until such time that the page information becomes invalid (see section 6.3.2). The values on EA[13:11] during column accesses and the DEAC command are the values latched during the ACTV command.

SBSRAM Interface

[0202]   As shown in Figure 35, the EMIF interfaces directly to industry standard synchronous burst SRAMs. This memory interface allows a high speed memory interface without some of the limitations of by SDRAM. Most notably, since SBSRAM are SRAM devices, random accesses in the same direction may occur in a single cycle. The SBSRAM interface may run at either the CPU clock speed or at ½ of this rate. The selection is made based on the setting of the SSCRT bit in the EMIF Global Control Register.
[0203]   The four SBSRAM control pins of Table 45 are latched by the SBSRAM on the rising SSCLK edge to determine the current operation. These signals are only valid if the chip select line for the SBSRAM is low. The /ADV signal is used to allow the SBSRAM device to generate addresses internally for interfacing to controllers which cannot provide addresses quickly enough. The EMIF does not need to use this signal because it can generate the address at the required rate.

55

TABLE 45.

EMIF SBSRAM Pins		
EMIF Signal	SBSRAM Signal	SBSRAM Function
SSADS	/ADS	Address Strobe.
/SSOE	/OE	Output Enable
/SSWE	/WE	Write Enable
SSCLK	CLK	SBSRAM Clock

Optimizing SBSRAM Accesses

[0204] SBSRAMs are latent by their architecture, meaning that read data follows address and control information. Consequently, the EMIF inserts cycles between read and write commands to ensure that no conflict exists on the ED [31:0] bus. The EMIF keeps this turn-around penalty to a minimum. The initial 2-cycle penalty is present when changing directions on the bus. In general, the rule is this; the first access of a burst sequence will incur a 2 cycle startup penalty.

SBSRAM Reads

[0205] Figure 36 shows a 16 word read of an SBSRAM. Every access strobes a new address into the SBSRAM, indicated by the /SSADS strobe low. The first access requires an initial startup penalty of 2 cycles; thereafter all accesses occur in a single SSCLK cycle.

SBSRAM Writes

[0206] Figure 37 shows a 6 word write of an SBSRAM. Every access strobes a new address into the SBSRAM. The first access requires an initial startup penalty of 2 cycles; thereafter all access can occur in a single SSCLK cycle.

Asynchronous Interface

[0207] The asynchronous interface offers users configurable memory cycle types, used to interface to a variety of memory and peripheral types; including, SRAM, EPROM, FLASH, as well as FPGA and ASIC designs.

[0208] Table 46 lists the asynchronous interface pins.

[0209] Figure 38 shows an interface to standard SRAM. Figure 39 shows an interface to FIFOs. Figure 40, Figure 41, and Figure 42 show interfaces to 8-, 16-, and 32-bit ROM. Although ROM may be interfaced at any of the CE spaces, it is often used at CE1 space because that space may be configured for widths of less than 32 bits.

TABLE 46.

EMIF Asynchronous Interface Pins	
EMIF Signal	Function
/AOE	Output Enable - active low during the entire period of a read access.
/AWE	Write Enable - active low during a write transfer strobe period.
/ARE	Read Enable - active low during a read transfer strobe period.
/ARDY	Ready input used to insert wait states into the memory cycle.

ROM Modes

[0210] The EMIF supports 8- and 16-bit wide ROMs access mode as selected by the MTYPE field in the EMIF CE Space Control Register. In reading data from these narrow width memory spaces, the EMIF packs multiple reads into one 32-bit wide value. This mode is primarily intended for word access to 8-bit and 16-bit ROM devices. Thus, the following restrictions apply:

Read operations will always read 32 bits, regardless of the access size or the memory width.

The address is shifted up appropriately to provide the correct address to the narrow memory. The shift amount is 1 for 16-bit ROM and 2 for 8-bit ROM. Thus, the high address bits are shifted out and accesses will wrap around if that /CE space spans the entire EA bus. Table 47 shows which address bits are present on the EA bus during an access to CE1 space for all possible asynchronous memory widths.

The EMIF always reads the lower addresses first and packs these into the LSBytes and packs subsequent accesses into the higher order bytes.

[0211]   Thus, the expected packing format in ROM is always little endian regardless of the value of the LENDIAN bit.
TABLE 47. Byte Address to EA Mapping for Asynchronous Memory Widths

width	EA2	EA3	EA4	EA5	EA6	EA7	EA8	EA9	EA10	EA11	EA12	EA13	EA14	EA15	EA16	EA17	EA18	EA19	EA20	EA21	EA22
x32	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
x16	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
x8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

8-Bit ROM

[0212] In 8-bit ROM mode, the address is left shifted by 2 to create a byte address on EA to access byte wide ROM. The EMIF always packs four consecutive bytes aligned on a 4-byte (byte address = 4N) boundary into a word access. The bytes are fetched in the following address order: 4N, 4N+1, 4N+2, 4N+3. Bytes are packed into the 32-bit word in the following little endian order from MSByte to LSByte: 4N+3, 4N+2, 4N+1, 4N.

16-Bit ROM

[0213] In 16-bit ROM mode, the address is left shifted by 1 to create a half-word address on EA to access 16-bit wide ROM. The EMIF always packs two consecutive half-words aligned on a 4-byte (byte address = 4N) boundary into a word access. The halfwords are fetched in the following address order: 4N, 4N+2. Halfwords are packed into the 32-bit word in the following little endian order from MSHalfword to LSHalfword: 4N+2, 4N.

Programmable ASRAM Parameters

[0214] The EMIF allows a high degree of programmability for shaping asynchronous accesses. The programmable parameters that allow this are:

Setup: The time between the beginning of a memory cycle (/CE low, address valid) and the activation of the read or write strobe.
Strobe: The time between the activation and deactivation of the read (/ARE) or write strobe (/AWE).
Hold: The time between the deactivation of the read or write strobe and the end of the cycle (which may be either an address change or the deactivation of the /CE signal)
Turn-around: The time between the disabling of device outputs (read data) and the driving of data into the memory device (write data).

[0215] These parameters are programmable in terms of CPU clock cycles via fields in the EMIF CE Space Control Registers (Figure 25). Separate setup, strobe, and hold strobe parameters are available for read and write accesses. The SETUP, HOLD, and STROBE fields represent actual cycle counts, in contrast to the SDRAM parameters which are the cycle counts - 1. The SETUP and STROBE fields have minimum count of 1. For SETUP and STROBE, a count of 0 will be treated as a count of 1. HOLD may be set to 0 cycles. The TA field, however, is specified in cycle count -1. Figure 43 will be used to illustrate the usage of these parameters.

Asynchronous Reads

[0216] Figure 43 illustrates three asynchronous reads. The first is a CE0 read with setup/strobe/hold set to 2/4/1. The second is the same but with additional strobe width due to an external inactive ready. The third is a CE1 read with setup/strobe/hold set to 4/5/3. An asynchronous read proceeds as follows:

At the beginning of the setup period

-    /CE becomes active low.
-    /AOE becomes active low.
-    /BE[3:0] becomes valid.
-    EA becomes valid.

At the beginning of a strobe period

-    /ARE is pulled active low.

At the beginning of a hold period

-    /ARE is pulled inactive high
-    Data is sampled on the CLKOUT1 rising edge concurrent with the beginning of the hold period (end of the strobe period), just prior to the /ARE low-to-high transition.

At the end of the hold period.

- /CE becomes inactive as long as another read access to the same /CE space is not scheduled for the next cycle.
- /AOE becomes inactive as long as another read access to the same /CE space is not scheduled for the next cycle.

Asynchronous Writes

[0217]   The first access in Figure 43, a CEO space write, illustrates an asynchronous write with a setup/strobe/hold of 3/6/3. An asynchronous write proceeds as follows.

At the beginning of the setup period

- /CE becomes active low.
- /BE[3:0] becomes valid.
- EA becomes valid.
- ED becomes valid.

At the beginning of a strobe period

- /AWE becomes inactive low.

At the beginning of a hold period

- /AWE becomes inactive high
- Data is sampled on the CLKOUT1 rising edge concurrent with the beginning of the hold period (end of the strobe period), just prior to the /AWE low-to-high transition.

At the end of the hold period.

- ED becomes tri-state only if as another write access to the same /CE space is NOT scheduled for the next cycle.
- /CE becomes inactive only if another write access to the same /CE space is NOT scheduled for the next cycle.
- If the next access is a read a cycle of turn around is inserted before AOE is enabled.

Turn-around

[0218]   Figure 43 illustrates usage of the programmable turn-around parameter. To prevent bus contention the EMIF inserts programmable turn-around cycles in two situations:

1. Between a read access to an asynchronous CE space followed by any access to a different CE space (whether asynchronous or not). In Figure 43, this occurs between the second CEO read and the CE1 read. In this example, the turn-around time is 2 cycles set by TA=1 in the EMIF CE0 Space Control Register.
2. Between a read access to an asynchronous CE space followed by a write access to the same CE space. A constant turn-around of 1 cycle is inserted between a write followed by a read to the same CE space. This is shown in Figure 43 between the CEO write and the first CEO read. A constant turn-around of 1 cycle is inserted between two consecutive accesses to different CE spaces from the same requester. This turn-around cycle occurs regardless of the memory types in the two CE spaces. The number of turn around cycles inserted is TA+1 as defined in the EMIF CE Space Control Register. During the turn-around period the affected asynchronous CE space's /CE and /AOE are inactivated to turn off any devices in that space still driving the bus. These cycles are only inserted in the case where the time between the consecutive accesses is less than TA+1 cycles.

Ready Input

[0219]   The second CEO space read in Figure 43 illustrates read operation. In addition to programmable access shaping, one may insert extra cycles into the strobe period by deactivating the /ARDY input. Ready operation works as follows:

Ready takes three CLKOUT1 cycles to cause a reaction on the bus. If /ARDY is low during a CLKOUT1 cycle, the strobe period will be extended three full CLKOUT1 cycles after the edge that recognizes /ARDY.
For every CLKOUT1 edge that ready is inactive low after the sampling period starts, the strobe width is extended

by one clock.

[0220]   In Figure 43, the second read is extended by two clocks due to a two cycle inactive /ARDY.

[0221]   Because of the delay in the effects of /ARDY, one must be careful when setup and strobe times are very small. For example, with strobe periods of 1, the /ARDY is sampled two cycle before the end of the setup period. In this case, if the setup period is 2, the /ARDY is sampled on the same edge that begins the access.

Hold Interface

[0222]   The EMIF responds to hold requests for the external bus. The hold handshake allows an external device and the EMIF to share the external bus. The handshake mechanism uses two signals:

1. /HOLD Asynchronous hold request input. The external device drives this pin low to request bus access. Hold is the highest priority request that the EMIF can receive during active operation. When the hold is requested, the EMIF stops driving the bus at the earliest possible moment, which may entail completion of the current accesses, device deactivation, and SDRAM bank deactivation. The external device must continue to drive /HOLD low for as long as it wants to drive the bus. The /HOLD input is internally synchronized to the CPU clock.
2. /HOLDA: Hold acknowledge output. The EMIF asserts this signal active low after it has placed its signal outputs in the high impedance state. The external device may then drive the bus as required. The EMIF places all outputs in the high impedance state with the exception of the clock outputs: CLKOUT1, CLKOUT2, SDCLK and SSCLK.

[0223]   There is no mechanism to ensure that the external device does not attempt to drive the bus indefinitely. The designer should be aware of system level issues, such as refresh, which may need to be performed. During host requests, the refresh counters within the EMIF continue to log refresh requests, however, no refresh cycles may be performed until bus control is granted back to the EMIF, by returning the /HOLD input to the inactive high level. A user may prevent an external hold by setting the NOHOLD bit in the EMIF Global Control Register.

Priority

[0224]   Table 48 illustrates the priority scheme that the EMIF will use in the case of multiple pending requests. The priority scheme may change if the DMA channel that is issuing a request through the DMA controller is of high priority. This mode is set in the DMA by setting the PRI bit in the DMA Channel Primary Control Register.

[0225]   Once a requester (note in this instance the refresh controller is considered a requester also) is prioritized and chosen, no new requests will be recognized until either the chosen requester stops making requests or a subsequent higher priority request occurs. In this case, all issued requests of the previous requester will be allowed to complete while the new requester starts making its requests.

[0226]   If the arbitrate bit of the EMIF global control register (Figure 24) is set (RBTR8 = 1), once a requester gains control of the EMIF it maintains control as long as the requester needs the EMIF, or if 8 word requests have occurred until a higher priority requester requests the EMIF. If a higher priority requester needs the EMIF, it will not get control until the current controller relinquishes control or until 8 word requests have completed. If the arbitrate bit is not set (RBTR8 = 0), a requester will maintain control of the EMIF as long as it needs the EMIF or until a higher priority requester requests the EMIF. The current controller will be interrupted by a higher priority requester, regardless of the number of requests which have occurred.

TABLE 48.

EMIF Prioritization of Requests		
Priority	Requester PRI=1	Requester PRI=0
Highest	External Hold Mode Register Set Urgent Refresh DMA DMC PMC	DMC PMC DMA

TABLE 48. (continued)

EMIF Prioritization of Requests		
Priority	Requester PRI=1	Requester PRI=0
Lowest	Trickle Refresh	

Clock Output Enabling

[0227] To reduce EMI radiation, the EMIF allows disabling (holding high) CLKOUT2, CLKOUT1, SSCLK, and SDCLK. This feature is performed via the CLK2EN, CLK1EN, SSCEN, and SDCEN bits in the EMIF SDRAM control register.

Power Down Operation

[0228] In power down 2 refresh is enabled. SSCLK, CLKOUT1, CLKOUT2 are held high. In power down 3 the EMIF acts as if it were in reset.

Emulation Halt Operation

[0229] The EMIF continues operating during emulation halts. Emulator accesses through the EMIF can have the effect of changing EMIF state and forcing startup penalties once the halt is stopped.

Boot

Overview

[0230] For proper device initialization, the device provides a variety of boot configurations. These configurations determine what actions the microprocessor 1 performs after device reset to prepare for initialization. These boot configurations, which are set by external input pins, determine:

The memory map the device selects. The memory map determines whether internal or external memory is mapped at address 0.
The type of external memory at address 0, (if external memory is mapped at address 0).
The boot process used to initialize the memory at address 0 before the CPU is allowed to run.

Device Reset

[0231] The external device reset 76 is an active low RESET-signal. While RESET- is active low the device is held in reset. During this period the device is initialized to the prescribed reset state. All tri-stateable outputs are tristated. All other outputs are returned to the state as described in the associated chapter. To allow reset is latched with the device CLKIN as well as the CPU clock. Thus, reset has minimum low time in terms of CLKIN as well as CPU clock (CLKOUT1) cycles. The precise timing requirements for device reset are described in the data sheet. The rising edge of RESET- starts the processor running with the prescribed boot configuration.

Boot Configuration

[0232] External pins BOOTMODE[4:0] determine the boot configuration for the part. The value of BOOTMODE[4:0] is latched with the rising edge of RESET-. These pins must be valid with the proper setup and hold to this edge. Refer to the data sheet for specific timing requirements. This boot configuration determines (Table 49):

The device memory map.
The type of memory at the reset location, address 0.
What bootload process is initiated at reset.

TABLE 49.

Boot Configurations			
BOOT MODE [4:0]	Memory Map	Memory At Address 0	Boot Process
00000	MAP 0	SDRAM 4 banks of 8-bit (SWID=0)	none
00001	MAP 0	SDRAM 2 banks of 16-bit (SWID=1)	none
00010	MAP 0	32-Bit Asynchronous w/Default Timing	none
00011	MAP 0	½ Rate SBSRAM	none
00100	MAP 0	lx Rate SBSRAM	none
00101	MAP 1	Internal	none
00110	MAP 0	External, Default Values	HPI
00111	MAP 1	Internal	HPI
01000	MAP 0	SDRAM 4 banks of 8-bit (SWID=0)	8-bit ROM w/default timings
01001	MAP 0	SDRAM 2 banks of 16-bit (SWID=1)	8-bit ROM w/default timings
01010	MAP 0	32-Bit Asynchronous w/Default Timing	8-bit ROM w/default timings
01011	MAP 0	½ Rate SBSRAM	8-bit ROM w/default timings
01100	MAP 0	lx Rate SBSRAM	8-bit ROM w/default timings
01101	MAP 1	Internal	8-bit ROM w/default timings
01110		Reserved	
01111		Reserved	
10000	MAP 0	SDRAM 4 banks of 8-bit (SWID=0)	16-bit ROM w/default timings
10001	MAP 0	SDRAM 2 banks of 16-bit (SWID=1)	16-bit ROM w/default timings
10010	MAP 0	32-Bit Asynchronous w/Default Timing	16-bit ROM w/default timings
10011	MAP 0	½ Rate SBSRAM	16-bit ROM w/default timings
10100	MAP 0	lx Rate SBSRAM	16-bit ROM w/default timings
10101	MAP 1	Internal	16-bit ROM w/default timings
10110		Reserved	
10111		Reserved	
11000	MAP 0	SDRAM 4 banks of 8-bit (SWID=0)	32-bit ROM w/default timings
11001	MAP 0	SDRAM 2 banks of 16-bit (SWID=1)	32-bit ROM w/default timings
11010	MAP 0	32-Bit Asynchronous w/Default Timing	32-bit ROM w/default timings
11011	MAP 0	½ Rate SBSRAM	32-bit ROM w/default timings
11100	MAP 0	1x Rate SBSRAM	32-bit ROM w/default timings
11101	MAP 1	Internal	32-bit ROM w/default timings
11110		Reserved	
11111		Reserved	

Memory Map

[0233]   There are two memory maps, MAP 0 and MAP 1 (Table 50). They differ in that MAP 0 has external memory mapped at address 0 and MAP 1 has internal memory mapped at address 0. Note the spaces allocated for internal peripherals represent space reserved for that purpose. The entire memory space reserved may not be populated. Refer to the appropriate discussion for the peripherals and internal herein.

TABLE 50.

Memory Map				
Address (Hex)	Range	Size (bytes)	Description Memory Map 0	Memory Map 1
00000000	003FFFFF	4M	External Memory Interface -CE Space 0	Internal Program RAM
00400000	00FFFFFF	12M		External Memory Interface -CE Space 0
01000000	013FFFFF	4M	External Memory Interface	
		4M	-CE Space 1	
01400000	017FFFFF	256K	Internal Program RAM	External Memory Interface -CE Space 1
		256K		
01800000	0183FFFF	256K	Internal Peripheral Bus -EMIF Registers	
01840000	0187FFFF	256K	Internal Peripheral Bus -DMA Controller Registers	
01880000	018BFFFF	256K	Internal Peripheral Bus -HPI Register	
018C0000	018FFFFF	256K	Internal Peripheral Bus -MCSP 0 Registers	
01900000	0193FFFF	256K	Internal Peripheral Bus -MCSP 1 Registers	
01940000	0197FFFF	256K	Internal Peripheral Bus -Timer 0 Registers	
01980000	019BFFFF	256K	Internal Peripheral Bus -Timer 1 Registers	
019C0000	019FFFFF	256K	Internal Peripheral Bus -Interrupt Selector Registers	
01A00000	01FFFFFF	6M	Internal Peripheral Bus - Reserved	
02000000	02FFFFFF	16M	External Memory Interface -CE Space 2	
03000000	03FFFFFF	16M	External Memory Interface -CE Space 3	
04000000	7FFFFFFF	2G-64M	Reserved	
80000000	803FFFFF	4M	Internal Data RAM	
80400000	FFFFFFFF	2G-4M	Reserved	

Memory at Address Reset Address

[0234] As shown in Table 49, the boot configuration determines the type of memory located at the reset address for processor operation, address 0. When the BOOTMODE pins select MAP 1, this memory is internal. When the device mode is in MAP 0, the memory is external. When external memory is selected, BOOTMODE also determines the type of memory at the reset address. Effectively, these options provide alternative reset values to the appropriate EMIF control registers.

Boot Processes

[0235] The boot process also is determined by the BOOTMODE pins as shown in Table 50. Three types of boot processes are available:

1. No Boot Process: The CPU simply starts running from the memory located at address 0. In the case of SDRAM, the CPU is held until SDRAM initialization completes.
2. ROM Boot Process: In the ROM boot process, the memory located at the beginning of CE- space 1 is copied to address 0 by the DMA. Although the boot process begins when the device is released from external reset, this transfer occurs while the CPU is held in reset internally. The amount of memory copied is 16K 32-bit words. This process also allows for selection of the width of the ROM. In this case, the EMIF can automatically assemble consecutive 8-bit bytes or 16-bit halfwords to form the 32-bit instruction words to be moved. These values are expected to be stored in little endian format in the external memory, typically a ROM device.

This transfer is automatically done by channel 0. The DMA channel is setup to perform an un-synchronized single-frame block transfer of 16K 32-bit words from the beginning of CE-space 1 to address 0. If MAP 0 is selected,

the EMIF postpones any DMA requests until it is fully configured by any alternative reset values. In the case of SDRAM, the EMIF postpones transfers until the SDRAM has been initialized. Note that the location of CE- space 1 depends on the memory map selected.

When the DMA_INT0 transitions active, indicating completion of the block transfer, the CPU is removed from reset and allowed to run from address 0. Note that the DMA_INT0 condition will not be latched by the CPU as it occurs while the CPU is still in reset. Also, the DMA_INT0 only wakes up the CPU from internal reset if the ROM boot process is selected. Note that after this condition, the BLOCK COND in the DMA Channel 0 Secondary Control Register remains set. This must be cleared for the user to receive subsequent DMA_INT0 transitions to the CPU and DMA.

3. HPI Boot Process: In HPI boot process, the CPU is held in reset while the remainder of the device is awoken from reset. During this period, an external host through the HPI may initialize the CPU's memory space as necessary, including external memory configuration registers. Once external memory has been configured as necessary, the host may then access any external sections it needs to complete initialization. Once the host is through with all necessary initialization, it writes a 1 to the DSPINT bit in the HPI control register (HPIC). This write causes an active transition on the DSPINT signal. This transition, in turn, causes the boot configuration logic to remove the CPU from its reset state. The CPU then begins running from address 0. The DSPINT condition is not latched by the CPU because it occurs while the CPU is still in reset. Also, the DSPINT only wakes up the CPU from internal reset if the HPI boot process is selected.

MCSP

[0236]   Referring now to Figure 44, a block diagram of MCSP 120, inventive aspects and advantageous features of the present invention will be described in detail. See Appendix A for an explanation of acronyms.

[0237]   The Multi-Channel Serial Port (MCSP) is an improvement on a known serial port interface found on the TMS320C2x, and 'C5x, Digital Signal Processor devices available from Texas Instruments Incorporated. A detailed description of this serial port is provided in U.S. Patent No._____ (application TI-19942) which is incorporated herein by reference. Like its predecessors the MCSP provides the following features:

1. Full-Duplex communication
2. Double buffered data registers which allow a continuous data stream.
3. Independent framing and clocking for receive and transmit.
4. Direct interface to industry standard Codecs, Analog Interface Chips (AICs), and other serially connected A/D and D/A devices.
5. External shift clock generation or an internal programmable frequency shift clock.

In addition, the MCSP has the following capabilities:

1. Direct interface to:

T1/E1 framers
MVIP and ST-BUS compliant devices
IOM-2 compliant devices
AC97 compliant devices. The necessary multi-phase frame synchronization capability is provided.
IIS compliant devices
SPI devices

2. Multi-channel transmit and receive of up to 128 channels.
3. A wider selection of data sizes including 8-, 12-, 16-, 20-, 24-, or 32-bit μ-Law and A-Law companding 8-bit data transfers with LSB or MSB first
4. Programmable polarity for both frame synchronization and data clocks
5. Highly programmable internal clock and frame generation.

[0238]   MCSP 120 consists of a data path and control path as shown in Figure 44. Seven external pins 121a-121g listed in Table 51 connect the control and data paths to external devices. The data is communicated to devices interfacing to the MCSP via the Data Transmit (DX) pin for transmit and the Data Receive (DR) pin for receive. Control information in the form of clocking and frame synchronization is communicated via CLKX, CLKR, FSX, and FSR. Processor 1 communicates to the MCSP via 32-bit wide control registers accessible via the internal peripheral bus 110. The CPU 10 or DMA 100/101 reads the received data from the Data Receive Register (DRR) and writes the data

to be transmitted to the Data Transmit Register (DXR). Data written to the DXR is shifted out to DX via the Transmit Shift Register (XSR). Similarly, receive data on the DR pin is shifted into RSR and copied into RBR. RBR is then copied to DRR which can be read by the CPU or DMA. This allows internal data movement and external data communications simultaneously. The remaining registers, which are accessible to CPU 10, configure the control mechanism of the MCSP. These registers are listed in Table 52. The control block consists of internal clock generation, frame synchronization signal generation, and their control, and multi-channel selection. This control block sends notification of important events to the CPU and DMA via four signals shown in Table 53.

TABLE 51.

MCSP Interface Signals		
Pin	I/O/Z	Description
CLKR	I/O/Z	Receive clock
CLKX	I/O/Z	Transmit clock
CLKS	I	External clock
DR	I	Received serial data
DX	O/Z	Transmitted serial data
FSR	I/O/Z	Receive frame synchronization
FSX	I/O/Z	Transmit frame synchronization

TABLE 52.

MCSP Registers			
Hex Byte Address			Register Name[2]
MCSP 0	MCSP 1	Acronym	
-	-	RBR	MCSP Receive Buffer Register
-	-	RSR	MCSP Receive Shift Register
-	-	XSR	MCSP Transmit Shift Register
018C0000	01900000	DRR	MCSP Data Receive[3] Register
018C0004	01900004	DXR	MCSP Data Transmit Register
018C0008	01900008	SPCR	MCSP Serial Port Control Register
018C000C	0190000C	RCR	MCSP Receive Control Register
018C0010	01900010	XCR	MCSP Transmit Control Register
018C0014	01900014	SRGR	MCSP Sample Rate Generator Register
018C0018	01900018	MCR	MCSP Multi-Channel Register
018C001C	0190001C	RCER	MCSP Receive Channel Enable Register
018C0020	01900020	XCER	MCSP Transmit Channel Enable Register
018C0024	01900024	PCR	MCSP Pin Control Register

[2] The RBR, RSR, and XSR are not directly accessible via the CPU or DMA.

[3] This register is read-only to the CPU and DMA.

TABLE 53.

MCSP CPU Interrupts and DMA Event Synchronization	
Interrupy Name	Description
RINT	Receive Interrupt to CPU
XINT	Transmit Interrupt to CPU
REVT	Receive Synchronization Event to DMA
XEVT	Transmit Synchronization Event to DMA

[0239] The serial port is configured via the 32-bit Serial Port Control Register (SPCR) and Pin Control Register as shown in Figure 45 and Figure 46, respectively. The SPCR and PCR contains MCSP status information and contains bits that can be configured for desired operation. The operation of each bit-field will be discussed in later sections.

[0240] Table 54 and Table 55 describe each bit field of the SPCR and PCR, respectively.

TABLE 54.

Serial Port Control Register (SPCR) Bit-Field Description	
Name	Function
DX STAT	DX pin status. Reflects value driven on to DX pin when selected as a general purpose output.
DR STAT	DR pin status. Reflects value on DR pin.
RIOEN XIOEN	Receive/Transmit General Purpose I/O Mode (R/X)IOEN=0, DR/DX pin is not a general purpose I/O (R/X)IOEN=1, DR/DX pin used as general purpose I/O
CLKSTP	Clock Stop Mode CLKSTP=000b, Clock Stop Mode Disabled CLKSTP=100b, Clock starts with rising edge without delay CLKSTP=101b, Clock starts with rising edge with delay CLKSTP=110b, Clock starts with falling edge without delay CLKSTP=111b, Clock starts with falling edge with delay CLKSTP=other, reserved
DLB	Digital Loop Back Mode DLB = 0, Digital loop back mode disabled DLB = 1, Digital loop back mode enabled
RJUST	Receive Sign-Extension and Justification Mode RJUST=00b, right-justify and zero-fill MSBs in DRR RJUST=01b, right-justify and sign-extend MSBs in DRR RJUST=10b, left-justify and zero-fill LSBs in DRR RJUST=11b, reserved
RINTM XINTM	Receive/Transmit Interrupt mode (R/X)INTM=00b, (R/X)INT driven by (R/X)RDY (R/X)INTM=01b, (R/X)INT generated by new block in multi-channel mode (R/X)INTM=10b, (R/X)INT generated by a new frame synchronization (R/X)INTM=11b, (R/X)INT generated by (R/X) SYNCERR
RSYNCERR XSYNCERR	Receive/Transmit Synchronization Error (R/X)SYNCERR=0, no synchronization error (R/X)SYNCERR=1, synchronization error detected by MCSP.
RFULL	Receive Shift Register (RSR) Full RFULL = 0, RBR is not full RFULL = 1, RBR is full and DRR is not read
XEMPTY-	Transmit Shift Register (XSR) Empty XEMPTY- = 0, XSR is empty XEMPTY- = 1, XSR is not empty
RRDY XRDY	Receiver/Transmitter Ready (R/X)RDY=0, receiver/transmitter is not ready. (R/X)RDY=1, receiver is ready with data to be read from DRR or transmitter is ready with data in DXR.
RRST- XRST-	Receiver/transmitter reset. This resets and enables the receiver/ transmitter. (R/X)RST- = 0, The serial port receiver/transmitter is disabled and in reset state. (R/X)RST- = 1, The serial port receiver/transmitter is enabled.

TABLE 55.

Pin Control Register (PCR) Bit-Field Description	
Name	Function
XIOEN RIOEN	Transmit/Receive General Purpose I/O Mode (R/X)IOEN=0, DR pin is not a general purpose input; DX pin is not a general purpose output (R/X)IOEN=1, DR pin is a general purpose input; DX pin is a general purpose output. The serial port pins do not perform serial port operation.
FSXM	Transmit Frame Synchronization Mode FSXM = 0, Frame synchronization signal derived from an external source FSXM = 1, Frame synchronization is determined by the Sample Rate Generator frame synchronization mode bit FSGM in the SRGR.
FSRM	Receive Frame Synchronization Mode FSRM = 0, Frame synchronization pulses generated by an external device. FSR is an input pin FSRM = 1, Frame synchronization generated internally by sample rate generator. FSR is an output pin except when GSYNC=1 in SRGR.
CLKS_ STAT	CLKS pin status. Reflects value driven on to CLKS pin when selected as a general purpose output.
DX_STAT	DX pin status. Reflects value driven on to DX pin when selected as a general purpose output.
DR_STAT	DR pin status. Reflects value on DR pin when selected as a general purpose input.
FSXP FSRP	Receive/Transmit Frame Synchronization Polarity FS(R/X)P = 0, Frame synchronization pulse FS(R/X) is active high FS(R/X)P = 1, Frame synchronization pulse FS(R/X) is active low
CLKXP	Transmit Clock Polarity CLKXP = 0, Transmit data sampled on rising edge of CLKX CLKXP = 1, Transmit data sampled on falling edge of CLKX
CLKRP	Receive Clock Polarity CLKRP = 0, Receive data sampled on falling edge of CLKR CLKRP = 1, Receive data sampled on rising edge of CLKR

[0241] The Receive and Transmit Control Registers (RCR and XCR) shown in Figure 47 and Figure 48 configure various parameters of receive and transmit operation respectively. The operation of each bit-field will be discussed in later sections. Table 56 describes each bit field of RCR and XCR.

TABLE 56.

Receive/Transmit Control Register (RCR/XCR) Bit-Field Description	
Name	Function
(R/X) PHASE	Receive/Transmit Phases (R/X)PHASE=0, single phase frame (R/X)PHASE=1, dual phase frame
(R/X)FRLEN(1 /2)	Receive/Transmit Frame Length 1/2
(R/X)WDLEN(1 /2)	Receive/Transmit Word Length 1/2
RCOMPAND XCOMPAND	Receive/Transmit Companding Mode. Modes other than 00b are only enabled when the appropriate (R/X)WDLEN is 000b, indicating 8-bit data. (R/X)COMPAND=00b, no companding, data transfer starts with MSB first. (R/X)COMPAND=01b, no companding, 8-bit data, transfer starts with LSB first. (R/X)COMPAND=10b, compand using μ-law for receive/transmit data. (R/X)COMPAND=11b, compand using A-law for receive/transmit data.

EP 0 901 081 A2

TABLE 56. (continued)

Receive/Transmit Control Register (RCR/XCR) Bit-Field Description	
Name	Function
RFIG XFIG	Receive/Transmit Frame Ignore (R/X)FIG = 0, Receive/Transmit Frame synchronization pulses after the first restarts the transfer. (R/X)FIG = 1, Receive/Transmit Frame synchronization pulses after the first are ignored.
RDATDLY XDATDLY	Receive/Transmit data delay (R/X)DATDLY=00b, 0-bit data delay (R/X)DATDLY=01b, 1-bit data delay (R/X)DATDLY=10b, 2-bit data delay (R/X)DATDLY=11b, reserved

[0242]   As shown in Figure 44, the receive operation is triple-buffered and transmit operation is double buffered. Receive data arrives on DR and is shifted into the RSR. Once a full word (8-, 12-, 16-, 20-, 24-, or 32-bit) is received, the RSR is always copied to the Receive Buffer Register, RBR. RBR is then copied to DRR unless DRR is not read by the CPU or DMA.

[0243]   Transmit data is written by the CPU or DMA to the DXR. If there is no data in the XSR, the value in the DXR is copied to the XSR. Otherwise, the DXR is copied to the XSR after the last bit of data in the DXR has been shifted out on DX. After transmit frame synchronization, the XSR begins shifting out the transmit data on DX.

[0244]   The serial port transmitter and receiver are independently reset by the RRST- and XRST- bits in the Serial Port Control register. If RRST- = XRST- = 0, the entire serial port is in reset state. A device reset 76 also places the serial port in the reset state. When device reset 76 is removed, RRST- =XRST- =0, keeping the entire serial port in the reset state. Table 57 shows the state of MCSP pins when the serial port is reset due to receiver/transmitter reset (XRST- = RRST- = 0) and due to device reset 76.

TABLE 57.

Reset State of MCSP Pins		
MCSP PINS	DEVICE RESET	MCSP RESET
Receiver Reset (RRST-=0)		
DR	Hi-Z	Hi-Z Input
CLKR	Hi-Z	Hi-Z if Input; CLKRP if Output
FSR	Hi-Z	Hi-Z if Input; FSRP(inactive state) if Output
CLKS	Hi-Z	Input should be pulled-up or pulled-down ONLY if transmitter is also reset or if the transmitter is configured with CLKSM=1 in SRGR.
Transmitter Reset (XRST-=0)		
DX	Hi-Z	Hi-Z Output
CLKX	Hi-Z	Hi-Z if Input; CLKXP if Output
FSX	Hi-Z	Hi-Z if Input; FSXP(inactive state) if Output
CLKS	Hi-Z	Input should be pulled-up or pulled-down ONLY if receiver is also reset or if the receiver is configured with CLKSM=1 in SRGR.

The following occurs when the MCSP is reset:

1) The state machine is reset to its initial state. This initial state includes resetting counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include XEMPTY-, XRDY, and XSYNCERR. (R/X)SYNCERR can also be cleared to zero by the user.
2) Activity in the corresponding section (transmitter/receiver) of the serial port stops. FS(R/X) is driven to its inactive state if they are outputs or are placed in a high impedance state (high impedance state) if they are configured as

inputs (external frame synchronization when FS(R/X)M=0). CLK(R/X) is set to the value of its polarity bit, CLK(R/X)P, if driven by the internal sample rate generator or placed in a high impedance state (high impedance state) if the transmit and receive clocks are treated as inputs to the MCSP. Lastly, the D(R/X) pins will be in high impedance state when the transmitter and/or receiver and/or the device is reset. One exception is that the internal sample rate generator clock, CLKG, runs as configured regardless of the reset state of the serial port. The frame sync signal FSG from the sample rate generator is driven to an inactive state (same as the value on the frame sync polarity bit, FS(R/X)P) to facilitate for FSR/FSX if FS(R/X)M=1 during reset.

3) When MCSP is reset due to device reset 76, all serial port pins are driven to the hi-Z state. When the serial port remains in reset state, but not the device, the DR and DX pins may be used as general purpose I/O as described in a later section.

[0245]   The serial port initialization procedure includes the following steps:

1) Set XRST-=RRST-=0 in SPCR.
2) Program only the MCSP configuration registers (and not the data registers) listed in Table 52 as required when the serial port is in reset state (XRST- = RRST- = 0).
3) Wait for two bit clocks. This is to ensure proper synchronization internally.
4) Set XRST-=RRST-=1 to enable the serial port. Note that the value written to the SPCR at this time should have only the reset bits changed to 1 and the remaining bit-fields should have the same value as in Step 2 above.

[0246]   Alternatively, on either write (Steps 1 and 4 above), the transmitter and receiver may be placed in or taken out of reset individually by only modifying the desired bit. Note that the necessary duration of the active-low period of XRST- or RRST- is at least two bit clocks (CLKR/CLKX) wide.

[0247]   Note: (a) The appropriate bit-fields in the serial port configuration registers SPCR, PCR, RCR, XCR, and SRGR should only be modified by the user when the affected portion of the serial port is in reset. (b) Data register DXR can be modified by the CPU or DMA only when the transmitter is not in reset (XRST-=1). (c) The multi-channel selection registers MCR, XCER, and RCER can be modified at any time as long as they are not being used by the current block in the multi-channel selection; see Section 0 for further details in the multi-channel mode case.

[0248]   For example, the following values in the control registers resets and configures the transmitter while the receiver is running:

SPCR=0x0030 0001	Transmitter reset, transmit interrupt (XINT toCPU) generated by XSYNCERR; receiver is running with RINT driven by RRDY.
PCR=0x0000 0A00	FSX determined by FSGM in SRGR, receive clock driven by external source, transmit clock continues to be driven by sample rate generator.
SRGR-0x2000 0001	CPU clock drives the sample rate generator clock (CLKG) after a divide-by-2. A DXR-to-XSR copy generated the transmit frame sync signal.
XCR=0x8421 0840	Dual phase frame; phase 1 has nine 16-bit words; phase2 has five 12-bit words, and 1-bit data delay
SPCR=0x0031 0001	Transmitter taken out of reset

[0249]   The Ready Status is indicated by RRDY and XRDY. RRDY and XRDY indicate the ready status of the MCSP receiver and transmitter, respectively. Writes and reads of the serial port may be synchronized by polling RRDY and XRDY, or by using the events to DMA (REVT and XEVT) or interrupts to CPU (RINT and XINT) that they generate. Note that reading the DRR and writing to DXR affects RRDY and XRDY.

[0250]   Receive Ready Status includes the following: REVT, RINT, and RRDY; RRDY=1 indicates that the RBR contents have been copied to the DRR and that the data can be read by the CPU or DMA. Once that data has been read by either the CPU or DMA, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST-=0), the RRDY is cleared to 0 to indicate no data has yet been received and loaded into DRR. RRDY directly drives the MCSP receive event to the DMA (REVT). Also, the MCSP receive interrupt (RINT) to the CPU may be driven by RRDY if RINTM=00b in the SPCR.

[0251]   Transmit Ready Status includes the following: XEVT, XINT, and XRDY: XRDY=1 indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset (XRST- transitions from 0 to 1), the XRDY also transitions from 0 to 1 indicating that the DXR is ready for new data. Once new data is loaded by the CPU or DMA, XRDY is cleared to 0. However, once

this data is copied from the DXR to the XSR, XRDY transitions again from 0 to 1. Now again, the CPU or DMA can write to DXR although XSR has not been shifted out on DX as yet. XRDY directly drives the transmit synchronization event to the DMA (XEVT). Also, the transmit interrupt (XINT) to the CPU may also be driven by XRDY if XINTM=00b in the SPCR.

[0252]   CPU Interrupts: are requested by (R/X)INT. The receive interrupt (RINT) and transmit interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring these interrupts. They are set by the receive/transmit interrupt mode bit-field, (R/X)INTM, in the SPCR.

1) (R/X)INTM=00b. Interrupt on every serial word by tracking the (R/X)RDY bits in the SPCR. The RRDY and XRDY bits were described previously.
2) (R/X)INTM=01b. Interrupt after every 16-channel block boundary (in multi-channel selection mode) has been crossed within a frame. In any other serial transfer case, this setting is not applicable and therefore no interrupts are generated. This is described in detail later.
3) (R/X)INTM=10b. Interrupt on detection of frame synchroniza-tion pulses. This generates an interrupt even when the transmitter/receiver is in reset. This is done by synchronizing the incoming frame sync pulse to the CPU clock and sending it to the CPU via (R/X)INT. This is described in detail later.
4) (R/X)INTM=11b. Interrupt on frame synchronization error.

[0253]   Note that if any of the other interrupt modes are selected, (R/X)SYNCERR may be read to detect this condition. Synchronization error is described in more detail later.
[0254]   Figure 49 shows typical operation of the MCSP clock and frame sync signals. Serial clocks CLKR, and CLKX define the boundaries between bits for receive and transmit respectively. Similarly, frame sync signals FSR and FSX define the beginning of a serial word. The MCSP allows configuration of various parameters for data frame synchro-nization. This can be done independently for receive and transmit which includes the following options:

1) Polarities of FSR, FSX, CLKX, and CLKR may be independently programmed.
2) A choice of single or dual-phase frames.
3) For each phase, the number of words is programmable.
4) For each phase, the number of bits per word is programmable.
    Subsequent frame synchronization may restart the serial data stream or be ignored.
5) The data bit delay from frame synchronization to first data bit can be 0-, 1-, or 2-bit delays.
6) Right or left-justification as well as sign-extension or zero-filling can be chosen for receive data.

[0255]   Frame and Clock Operation will now be described. Receive and transmit frame sync pulses can be generated either internally by the sample rate generator (described later) or driven by an external source. This can be achieved by programming the mode bit, FS(R/X)M, in the PCR. FSR is also affected by the GSYNC bit in the SRGR (described later). Similarly, receive and transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLK(R/X)M, in the PCR.
[0256]   When FSR and FSX are inputs (FSXM=FSRM=0, external frame sync pulses), the MCSP detects them on the internal falling edge of clock, CLKR_int, and CLKX_int respectively (See Figure 79). The receive data arriving at DR pin is also sampled on the falling edge of CLKR_int. Note that these internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the MCSP.
[0257]   When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X)_int. Similarly data on DX pin is output on the rising edge of CLKX_int.
[0258]   FSRP, FSXP, CLKRP, and CLKXP configure the relative polarities of the external FSR, FSX, CLKR, and CLKX signals as shown in Table 55. All frame sync signals (FSR_int, FSX_int) internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to MCSP), and FSRP=FSXP=1, the external active low frame sync signals are inverted before being sent to the receiver (FSR_int) and transmitter (FSX_int). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC=0) is selected, the internal active high sync signals are inverted if the polarity bit FS(R/X)P=1, before being sent to the FS(R/X) pin. Figure 79 shows this inversion using XOR gates.
[0259]   On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Note that data is always transmitted on the rising edge of CLKX_int. If CLKXP=1, and external clocking is selected (CLKXM=0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP=1, and internal clocking selected (CLKXM=1 and CLKX is an output pin), the internal (rising-edge triggered) clock, CLKX_int, is inverted before being sent out on the CLKX pin.
[0260]   Similarly, on the transmitter side, the configuration is so that the receiver can reliably sample data that is

clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. Note that the receive data is always sampled on the falling edge of CLKR_int. Therefore, if CLKRP=1 and external clocking is selected (CLKRM=0 and CLKR is an input pin), the external rising edge triggered input clock on CLKR is inverted to a falling-edge before being sent to the receiver. If CLKRP=1, and internal clocking is selected (CLKRM=1), the internal falling-edge triggered clock is inverted to a rising-edge before being sent out on the CLKR pin.

[0261]   Note that in a system where the same clock (internal or external) is used to clock the receiver and transmitter, CLKRP=CLKXP. The receiver uses the opposite edge as the transmitter to guarantee valid setup and hold of data around this edge. Figure 50 shows how data clocked by an external serial device using a rising edge may be sampled by the MCSP receiver with the falling edge of the same clock.

[0262]   Frame Synchronization Phases will now be described. Frame synchronization is used to indicate the beginning of a transfer on the MCSP. The data stream following frame synchronization may have two phases, phase 1 and phase 2. The number of phases in a data stream can be selected by the phase bit, (R/X)PHASE, in the RCR and XCR. The number of words per frame and bits per word can be independently selected for each phase via (R/X)FRLEN(1/2) and (R/X)WDLEN(1/2) respectively. Figure 51 shows an example of a frame where the first phase consists of 2 words of 12 bits each followed by a second phase of 3 words of 8 bits each. Note that the entire bit stream in the frame is contiguous. There are no gaps either between words or between phases.

[0263]   Table 58 shows the bit-fields in the receive/transmit control register (RCR/XCR) that control the number of words/frame and bits/word for each phase for both the receiver and transmitter. The maximum number of words per frame is 128, and the number of bits per word can be 8-, 12-, 16-, 20-, 24-, or 32-bits.

Table 58.

RCR/XCR Bit-fields Controlling Words/Frame and Bits/Word			
Serial Port MCSP0/1	Frame Phase	RCR/XCR Bit-field Words/Frame	Control Bits/Word
Receive	1	RFRLEN1	RWDLEN1
Receive	2	RFRLEN2	RWDLEN2
Transmit	1	XFRLEN1	XWDLEN1
Transmit	2	XFRLEN2	XWDLEN2

[0264]   Frame Length, (R/X)FRLEN(1/2), will now be described, with reference to Table 59. Frame length can be defined as the number of serial words (8-, 12-, 16-, 20-, 24-, or 32-bit) transferred per frame. It corresponds to the number of words or logical time slots or channels per frame synchronization signal. The 7-bit (R/X)FRLEN(1/2) field in the (R/X)CR supports up to 128 words per frame as shown in Table 59. Note that (R/X)PHASE=0 represents a single phase data frame and a (R/X)PHASE=1 selects a dual phase for the data stream. Note that for a single phase frame, FRLEN2 is a don't care. The user is cautioned to program the frame length fields with w minus 1, where w represents the number of words per frame. For the example in Figure 51, (R/X)FRLEN1=1 or 0000001b and (R/X)FRLEN2=2 or 0000010b.

TABLE 59.

MCSP Receive/Transmit Frame Length 1/2 Configuration			
(R/X)PHASE	(R/X)FRLE N1	(R/X)FRL EN2	Frame Length
0	0 ≤ n ≤ 127	X	Single Phase Frame; (n+1) words per frame
1	0 ≤ n ≤ 127	0 ≤ m ≤ 127	Dual Phase Frame; (n+1) plus (m+1) words per frame

[0265]   Word Length, (R/X)WDLEN(1/2), will now be described. The 8-bit (R/X)WDLEN(1/2) fields in the receive/transmit control register determine the word length in bits per word for the receiver and transmitter for each phase of the frame as shown in Table 58. Table 60 shows how the value of these fields selects particular word lengths in bits. For the example in Figure 51, (R/X)WDLEN1=001b, and (R/X)WDLEN2=000b. Note that if (R/X)PHASE=0 indicating a single phase frame, R/X)WDLEN2 is not used by the MCSP and its value is a don't care.

TABLE 60.

MCSP Receive/Transmit Word Length Configuration	
(R/X)WDLEN(1/2)	MCSP Word Length (bits)
000	8
001	12
010	16
011	20
100	24
101	32
110	reserved
111	reserved

[0266]   Data Packing using Frame Length and Word Length will now be described. The frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single phase frame as shown in Figure 52. In this case:

(R/X)FRLEN1=0000011b, 4-word frame
(R/X)PHASE=0, single phase frame
(R/X)FRLEN2=X
(R/X)WDLEN1=000b, 8-bit words

[0267]   In this case, four 8-bit data elements are transferred to and from the MCSP by the CPU or DMA. Thus, four reads of DRR and four writes of DXR are necessary for each frame.

[0268]   The example in Figure 52 can also be treated as a data stream of a single phase frame consisting of one 32-bit data word as shown in Figure 53. In this case:

(R/X)FRLEN1=0b, 1-word frame
(R/X)PHASE=0, single phase frame
(R/X)FRLEN2=X
(R/X)WDLEN1=101b, 32-bit words.

[0269]   In this case, one 32-bit data word is transferred to and from the MCSP by the CPU or DMA. Thus, one read of DRR and one write of DXR is necessary for each frame. This results in only one-fourth the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

[0270]   Data Delay, (R/X)DATDLY, will now be described. The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay. RDATDLY and XDATDLY specify the data delay for reception and transmission, respectively. The range of programmable data delay is zero to two bit-clocks ((R/X)DATDLY=00b -10b) as described in Table 56 and shown in Figure 54. Typically 1-bit delay is selected as data often follows a one-cycle active frame sync pulse.

[0271]   Normally, frame sync pulse is detected or sampled with respect to an edge of serial clock CLK(R/X)_int (described earlier). Thus, on the following cycle or later (depending on data delay value), data may be received or transmitted. However, in the case of zero-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle. For reception, this problem is solved as receive data is sampled on the first falling edge of CLKR_int where an active high FSR_int is detected. However, data transmission must begin on the rising edge of CLKX_int clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in the XSR and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX_int, going active high, and immediately starts driving the first bit to be transmitted on the DX pin.

[0272]   Another common mode is a data delay of two. This configuration allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after one-bit delay and data appears after 2-bit delay), the serial port

essentially discards the framing bit from the data stream as shown in Figure 55. In transmission, by delaying the first transfer bit, the serial port essentially inserts a blank period (high impedance period) where framing bit should be. Here, it is expected that the framing device inserts its own framing bit or that the framing bit is generated by another device.

[0273] A Multi-phase Frame Example will now be described. Figure 56 shows an example of the Audio Codec '97 (AC97) standard which uses the dual phase frame feature. The first phase consists of a single 16-bit word. The second phase consists of 12 20-bit words. The phases are configured as follows:

    (R/X)PHASE=1b, dual phase frame
    (R/X)FRLEN1=0b, 1 word per frame in phase 1
    (R/X)WDLEN1=010b, 16-bits per word in phase 1
    (R/X)FRLEN2=0001011b, 12 words per frame in phase 2
    (R/X)WDLEN2=011b, 20-bits per word in phase 2
    CLK(R/X)P=0, receive data sampled on falling edge of CLKR_int; transmit data clocked on rising edge of CLKX_int.
    FS(R/X)P=0, active high frame sync signals
    (R/X)DATDLY=01b, data delay of one bit-clock

[0274] Figure 56 shows the timing of AC97 near frame synchronization. First notice that the frame sync pulse itself overlaps the first word. In MCSP operation, the inactive to active transition of the frame synchronization signal actually indicates frame synchronization. For this reason, frame synchronization may be high an arbitrary number of bit clocks. Only after the frame synchronization is recognized to have gone inactive and then active again is the next frame synchronization recognized.

[0275] Also notice in Figure 57, there is one-bit data delay. Notice that regardless of the data delay, transmission can occur without gaps. The last bit of the previous (last) word in phase 2 is immediately followed by the first data bit of the first word in phase 1 of the next data frame.

[0276] MCSP Standard Operation Mode will now be described. Depending on the mode of operation, there can be periods of serial port inactivity between packets or transfers. In this case, the receive and transmit frame synchronization pulse occurs for every serial word transfer. When the MCSP is not in reset state and has been configured for the desired operation, a serial transfer can be initiated by programming (R/X)PHASE=0 for a single phase frame with required number of words programmed in (R/X)FRLEN1. The number of words can range from 1 to 128 ((R/X)FRLEN1=0x0 to 0x7F). The required serial word length is set in the (R/X)WDLEN1 field in the (R/X)CR. If dual phase is required for the transfer, RPHASE=1, (R/X)FRLEN(1/2) can be set to any value between 0x0 to 0x7F which represents 1 to 128 words.

[0277] Figure 58 shows an example of a single phase data frame comprising one 8-bit word. Since the transfer is configured for one data bit delay, the data on the DX and DR pins are available one bit clock after FS(R/X) goes active. This figure as well as all others in this section make the following assumptions:

    (R/X)FRLEN1=0b, 1 word per frame
    (R/X)PHASE=0, single phase frame
    (R/X)FRLEN2=X, don't care
    (R/X)WDLEN1=000b, 8-bit word
    (R/X)CLKP=0, receive data clocked on falling edge; transmit
    data clocked on rising edge
    FS(R/X)P=0, active high frame sync signals
    (R/X)DATDLY=01b, one-bit data delay

[0278] Figure 59 shows an example of serial reception. Once receive frame synchronization (FSR_int) transitions to its active state, it is detected on the first falling edge of CLKR_int of the receiver. The data on the DR pin is then shifted into the receive shift register (RSR) after the appropriate data delay as set by RDATDLY. The contents of RSR is then shifted to RBR. RRDY is activated on every RBR-to-DRR copy to indicate that the receive data register (DRR) is ready with the data to be read by the CPU or DMA. RRDY is deactivated when the DRR is read by the CPU or DMA.

[0279] Figure 60 shows an example of serial transmission. Once transmit frame synchronization occurs, the value in the transmit shift register, XSR, is shifted out and driven on the DX pin after the appropriate data delay as set by XDATDLY. XRDY is activated on every DXR-to-XSR copy indicating that the data transmit register (DXR) is written with the next data to be transmitted. XRDY is deactivated when the DXR is written by the CPU or DMA.

[0280] Figure 61 shows the MCSP operating at maximum packet frequency. The total number of words in a frame, whether single phase or dual phase, may be called the serial transfer packet. The packet frequency is determined by the period between frame synchronization signals:

$$\text{Packet Frequency} = \frac{\text{(Bit - Clock Frequency)}}{\text{(Number of Bit Clocks Between Frame Sync Signals)}}$$

[0281] The packet frequency may be increased by decreasing the distance between frame synchronization signals in bit clocks (limited only by the number of bits per packet). As the packet transmit frequency is increased, the inactivity period between the data packets for adjacent transfers decreases to zero. The minimum distance between frame synchronization is the number of bits transferred per packet. This distance also defines the maximum packet frequency:

$$\text{Maximum Packet Frequency} = \frac{\text{(Bit - Clock Frequency)}}{\text{(Number of Bits Per Frame or Packet)}}$$

[0282] At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits. If there is a one-bit data delay as shown, the frame synchronization pulse overlaps the last bit transmitted in the previous packet.

[0283] Effectively, this permits a continuous stream of data, and, thus the frame synchronization pulses are essentially redundant. Theoretically, then, only an initial frame synchronization pulse is required to initiate the multi-packet transfer. The MCSP does support operation of the serial port in this fashion by ignoring the successive frame sync pulses. Data is clocked in to the receiver or clocked out of the transmitter on every clock . The frame ignore bit, (R/X)FIG, in the (R/X)CR can be programmed to ignore the successive frame sync pulses until the desired frame length or number of words is reached. This is explained later.

[0284] Frame Synchronization Ignore Mode will now be described. The MCSP can be configured to ignore transmit and receive frame synchronization pulses. The (R/X)FIG bit in the (R/X)CR can be programmed to zero and not ignore frame sync pulses, or, be set to one and ignore frame sync pulses. This way the user can use (R/X)FIG bit to either pack data or ignore unexpected frame sync pulses. Data packing is explained in the following paragraph; and MCSP operation on unexpected frame sync pulses in the next paragraph.

[0285] Data Packing using Frame Sync Ignore Bits will now be described. One method of changing the word length and frame length to simulate 32-bit serial word transfers was described with reference to Figure 52, thus requiring much lesser bus bandwidth. This example worked when there were multiple words per frame. Now consider the case of the MCSP operating at maximum packet frequency as shown in Figure 62. Here, each frame only has a single 8-bit word. This stream takes one read and one write transfer for each 8-bit word. Figure 63 shows the MCSP configured to treat this stream as a continuous stream of 32-bit words. In this example (R/X)FIG is set to 1 to ignore subsequent frames after the first. Here, only one read and one write transfer is needed every 32-bits. This configuration effectively reduces the required bus bandwidth to one-fourth.

[0286] Frame Sync Ignore and Unexpected Frame Sync Pulses will now be described. The previous section explained how frame ignore bits can be used to pack data and efficiently use the bus bandwidth. (R/X)FIG bit can also be used to ignore unexpected frame sync pulses. Unexpected frame sync pulses are those that occur at a time when they are not expected. Thus, any frame sync pulse which occurs one or more bit clocks earlier than the programmed data delay ((R/X)DATDLY) is deemed as unexpected. Setting the frame ignore bits to one causes the serial port to ignore these unexpected frame sync signals.

[0287] In reception, if not ignored (RFIG=0), an unexpected FSR pulse will discard the contents of RSR in favor of the new incoming data. Therefore if RFIG=0, an unexpected frame synchronization pulse aborts the current data transfer, sets RSYNCERR in the SPCR to 1, and begins the transfer of a new data word. This is described in more detail later. When RFIG=1, reception continues, ignoring the unexpected frame sync pulses.

[0288] If (R/X)FIG is set to zero, these frame sync pulses are not ignored. In transmission, if not ignored (XFIG=0), an unexpected FSX pulse will abort the present transmission, cause one data packet to be lost, sets the XSYNCERR in the SPCR to 1, and initiates a new transmission. This is described in more detail later. When XFIG=1, normal transmission continues with unexpected frame sync signals ignored.

[0289] Figure 64 shows an example wherein word B is interrupted by an unexpected frame sync pulse when (R/X)FIG=0. In the case of reception, the reception of B is aborted (B is lost), and a new data word (C in this example) is received after the appropriate data delay. This condition is a receive synchronization error and thus sets the RSYNCERR field. However, for transmission, the transmission of B is aborted, and the same data (B) is retransmitted after the appropriate data delay. This condition is a transmit synchronization error and thus sets the XSYNCERR field. Synchronization errors are discussed later. In contrast, Figure 65 shows MCSP operation when unexpected frame synchronization signals are ignored by setting (R/X)FIG=1. Here, the transfer of word B is not affected by an unexpected frame synchronization.

[0290] Serial Port Exception Conditions will now be described. There are five serial port events that may constitute a system error, as follows:

1) Receive Overrun (RFULL=1). This occurs when RBR is full and the previous received data in the DRR has not been read (RRDY=1) by the CPU or DMA. The contents of RBR will not be transferred to DRR until DRR has been read.

2) Unexpected Receive Frame Synchronization (RSYNCERR=1). This can occur during reception when RFIG=0 and an unexpected frame sync pulse occurs. An unexpected frame sync pulse is defined as that which occurs RDATDLY minus one or more bit-clocks earlier than the first bit of the next associated word. This causes the current data reception to abort and restart. If new data has been copied into the RBR from RSR since the last RBR-to-DRR copy, this new data in RBR will be lost. This is because no RBR-to-DRR copy occurs as the reception has been restarted.

3) Transmit Data Overwrite. Here the user overwrites data in the DXR before it is copied to the XSR. The data previously in the DXR is never transferred on DX since it never got copied to the XSR.

4) Transmit Empty (XEMPTY- = 0). If a new frame synchroniza-tion signal arrives before new data is loaded into the DXR, the old data in the DXR will be sent again. This will continue for every new frame sync signal that arrives on the FSX pin until the DXR is loaded with new data.

5) Unexpected Transmit Frame Synchronization (XSYNCERR=1). This can occur during transmission when XFIG=0 and an unexpected frame sync pulse occurs. Again, an unexpected frame sync pulse is defined as that which occurs XDATDLY minus one or more bit-clocks earlier than the first bit of the next associated word. This causes the current data transmission to abort and restart the current transfer. If new data had been written to the DXR since the last DXR-to-XSR copy, the current value in the XSR will be lost. Otherwise, the previous value will be retransmitted.

[0291]   These events are described in more detail in the following paragraphs.

[0292]   Reception with Overrun, RFULL will now be described. RFULL=1 in the SPCR indicates that the receiver has experienced overrun. Overrun occurs when: RBR is full, and DRR has not been read since the last RBR-to-DRR transfer (RRDY=1).

[0293]   During this condition, any new data input on DR pin will still be shifted into RSR. Once a complete word is shifted into RSR, and RFULL=1, the receiver halts and waits for DRR to be read and does not perform a RBR-to-DRR transfer. At this time any data arriving on the DR pin will be lost. This data loss is because completion of reception of a serial word triggers an RSR-to-DRR transfer only when RRDY=0. Note that after the receive portion starts running from reset, a minimum of three words must be received before RFULL is set. That is because there was no last RBR-to-DRR transfer before the first word.

[0294]   Any one of the following events clears the RFULL bit to 0 and allows subsequent transfers to be read properly: 1.) Reading DRR; 2.) Resetting the receiver (RRST-=0) or the device. Another frame synchronization is required to restart the receiver.

[0295]   Figure 66 shows a receive overrun condition. Because serial word A is not read before the completion of the reception of serial word B, B is never transferred to the DRR. Thus, at the end of reception of B, RFULL is set. When the DRR is finally read, the value A is still in the DRR. Because the read of the DRR occurs after the serial word C has started transfer, C is never shifted into the RSR.

[0296]   Figure 67 shows the case where RFULL is set, but the overrun condition is averted by reading the contents of DRR before the next serial word. This ensures an RBR-to-DRR copy of data B occurs before the next serial word is transferred from RSR to RBR.

[0297]   Unexpected Receive Frame Synchronization, RSYNCERR, will now be described. Figure 68 shows the decision tree that the receiver uses to handle all incoming frame synchronization pulses. The diagram assumes that the receiver has been started, RRST-=1. Unexpected frame sync pulses can originate from an external source or from the internal sample rate generator. Any one of four cases can occur:

Case 1: FSR_int pulses during receive overrun. The transfer of word C in Figure 66 is an example of this. These frame sync pulses are ignored and receiver stays halted.

Case 2: Unexpected FSR_int pulses with RFIG=1. This case is discussed with reference to Figure 65. Here, receive frame sync pulses are ignored and the reception continues.

Case 3: Normal serial port reception. Note that there are three possible reasons why a receive might NOT be in progress:

1) This FSR is the first after RRST-=1.

2) This FSR is the first after DRR is read clearing a RFULL condition.

3) The serial port is in the inter-packet intervals. Thus, at maximum packet frequency, frame synchronization can still be received RDATDLY bit clocks before the first bit of the associated word. Alternatively, an unexpected frame sync pulse is detected when it occurs at or before RDATDLY minus one bit clock before the last bit of

the previous word is received on DR pin.

For this case, reception continues normally since these are not unexpected frame sync pulses.

4) Case 4: Unexpected receive frame synchronization with RFIG=0 (unexpected frame not ignored). This case was shown in Figure 24 for maximum packet frequency. Figure 69 shows this case during normal operation of the serial port with inter-packet intervals. In both cases, RSYNCERR bit in the SPCR is set. RSYNCERR can be cleared only by receiver reset or by the user writing a 0 to this bit in the SPCR. Note that if RINTM=11b in the SPCR, RSYNCERR drives the receive interrupt (RINT) to the CPU.

[0298] Transmit with Data Overwrite will now be described. Figure 70 depicts what happens if the data in DXR is overwritten before being transmitted. Initially, the programmer loaded the DXR with data C. A subsequent write to the DXR overwrites C with D before it is copied to the XSR. Thus, C is never transmitted on DX. The CPU can avoid data overwrite by polling XRDY before writing to DXR or by waiting for an XINT programmed to be triggered by XRDY (XINTM=OOb). The DMA can avoid overwriting by write synchronizing data transfers with XEVT.

[0299] XEMPTY- indicates whether the transmitter has experienced under-flow. Any of the following conditions causes XEMPTY- to become active (XEMPTY- = 0):

1) Under-flow during transmission. DXR has not been loaded since the last DXR-to-XSR copy, and all bits of the data word in the XSR have been shifted out on DX.
2) The transmitter is reset (XRST- = 0 or device is reset) and then restarted.

[0300] When XEMPTY- = 0, the transmitter halts and places DX in high-impedance until the next frame synchronization. XEMPTY- is deactivated (XEMPTY-= 1) after DXR is loaded by either the CPU or DMA. If DXR is reloaded after a halt due to under-flow, the device begins transmission again. In the case of internal frame generation, the transmitter re-generates a single FSX int initiated by a DXR-to-XSR copy (FSXM=1 in the XCR and FSGM=0 in SRGR). Otherwise, the transmitter waits for the next frame synchronization.

[0301] Figure 71 depicts a transmit under-flow condition. After B is transmitted, the programmer fails to reload the DXR before the subsequent frame synchronization. Thus, B is again transmitted on DX. Figure 72 shows the case of writing to DXR just before a transmit under-flow condition that would otherwise occur. After B is transmitted, C is written to DXR before the next transmit frame sync pulse occurs so that C is successfully transmitted on DX, thus averting a transmit empty condition.

[0302] Unexpected Transmit Frame Synchronization, XSYNCERR, will now be described. Figure 73 shows the decision tree that the transmitter uses to handle all incoming frame synchronization signals. The diagram assumes that the transmitter has been started, XRST-=1. Any one of three cases can occur:

Case 1: Unexpected FSX_int pulses with XFIG=1. This case is discussed with reference to Figure 65.
Case 2: Normal serial port transmission was discussed earlier. Note that there are two possible reasons why a transmit might NOT be in progress:

1) This FSX_int pulse is the first after XRST-=1.
2) The serial port is in the inter-packet intervals. Thus, if operating at maximum packet frequency, frame synchronization can still be received XDATDLY bit clocks before the first bit of the associated word.

Case 3: Unexpected transmit frame synchronization with XFIG=0.

[0303] The case for subsequent frame synchronization with XFIG=0 at maximum packet frequency is shown in Figure 64. Figure 74 shows the case for normal operation of the serial port with inter-packet intervals. In both cases, XSYN-CERR bit in the SPCR is set. XSYNCERR can only be cleared by transmitter reset or by the user writing a 0 to this bit in the SPCR. Note that if XINTM=11b in the SPCR, XSYNCERR drives the receive interrupt (XINT) to the CPU. Note further that the XSYNCERR bit in the SPCR is a read/write bit. Therefore, writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

[0304] Receive Data Justification and Sign-Extension, RJUST, will now be described. RJUST in the SPCR selects whether data in the RBR is right or left justified (with respect to the MSB) in the DRR. If right-justification is selected RJUST further selects whether the data is sign-extended or zero-filled. Table 61 shows the effect various modes of RJUST have on an example 12-bit receive data value 0xABC.

TABLE 61.

Use of RJUST Field with 12-Bit Example Data 0xABC			
RJUST	Justification	Extension	Value in DRR
00	right	zero-fill MSBs	0x00000ABC
01	right	sign-extend MSBs	0xFFFFFABC
10	left	zero-fill LSBs	0xABC00000
11	reserved	reserved	reserved

[0305] Operation of the μ-LAW/A-LAW Companding Hardware (R/X)COMPAND, will now be described. Companding (COMpress and exPAND) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred as A-law. The specification for μ-law and A-law log PCM is part of the CCITT G.711 recommendation. A-law and μ-law allows 13-bits and 14-bits of dynamic range, respectively. Any values outside this range will be set to the most positive or most negative value.

[0306] The μ-law and A-law formats encode this data into 8-bit code words. Thus, as companded data is always 8-bit wide, the appropriate (R/X)WDLEN(1/2) must be set to 0, indicating 8-bit wide serial data stream. If either phase of the frame does not have 8-bit word length, then companding is automatically disabled for that portion (transmit or receive) and phase.

[0307] When companding is used, transmit data is encoded according to specified companding law, and receive data is decoded to 2's complement format. Companding is enabled and the desired format selected by appropriately setting (R/X)COMPAND in the (R/X)CR as shown in Table 56. Compression occurs during the process of copying data from the DXR-to-XSR and from the RBR-to-DRR as shown in Figure 75.

[0308] For transmit data to be compressed, it should be a 16-bit left justified data, say LAW16. The value can be either 13- or 14-bits depending on the companding law. This is shown in Figure 76.

[0309] For reception, the 8-bit compressed data in RBR is expanded to a left-justified 16-bit data, LAW16. This can be further justified to a 32-bit data by programming the RJUST field in the SPCR as shown in Table 62.

TABLE 62.

Justification of Expanded Data (LAW16)		
RJUST	DRR	
	31 16	15 0
00	0	LAW16
01	sign	LAW16
10	LAW16	0
11	reserved	

[0310] If the MCSP is otherwise unused, the companding hardware can compand internal data. This can be used to:

a) Convert linear to the appropriate μ-law or A-law format.

b) Convert μ-law or A-law to the linear format.

c) To observe the quantization effects in companding by transmitting linear data, and compressing and re-expanding this data. This is only useful if both XCOMPAND and COMPAND enable the same companding format.

[0311] Figure 77 shows two methods by which the MCSP can compand internal data.

1) When both the transmit and receive sections of the serial port are reset, the DRR and DXR are internally connected through the companding logic. Values from the DXR are compressed as selected by XCOMPAND and then expanded as selected by RCOMPAND. Data is available within four CPU clocks after being written. The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU and DMA to control the flow.

2) The MCSP is enabled in digital loop back mode with companding appropriately enabled by RCOMPAND and

XCOMPAND. Receive and transmit interrupts (RINT when RINTM=O and XINT when XINTM=0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

[0312] Normally, all transfers on the MCSP are sent and received with the MSB first. However, certain 8-bit data protocols (that don't use companded data) require the LSB to be transferred first. By setting the (R/X)COMPAND=01b in the (R/X)CR, the bit ordering of 8-bit words are reversed (LSB first) before being sent to the serial port. Similar to companding, this feature is only enabled if the appropriate (R/X)WDLEN(1/2) is set to 0, indicating 8-bit words to be transferred serially. If either phase of the frame does not have 8-bit word length, then LSB first ordering is automatically disabled for that portion (transmit or receive) and phase.

[0313] Programmable Clock and Framing will now be described. The MCSP has several means of selecting clocking and framing for both the receiver and transmitter. Clocking and framing can be sent to both portions by the sample rate generator. Both portions can select external clocking and/or framing independently. Figure 38 shows a block diagram of the clock and frame selection circuitry. The features enabled by this logic are explained in the following paragraphs.

[0314] The sample rate generator is composed of a three stage clock divider that allows programmable data clocks (CLKG) and framing signals (FSG) as shown in Figure 79. CLKG and FSG are MCSP internal signals that can be programmed to drive receive and/or transmit clocking (CLKR/X) and framing (FSR/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source. The three stages compute:

1) Clock divide down (CLKGDV): The number of input clocks per data bit clock.
2) Frame period divide down (FPER): The frame period in data bit clocks.
3) Frame width count down (FWID): The width of an active frame pulse in data bit clocks.

[0315] In addition, a frame pulse detection and clock synchronization module allows synchronization of the clock divide down with an incoming frame pulse.

[0316] The Sample Rate Generator Register (SRGR) shown in Figure 80 and Table 63 controls the operation of various features of the sample rate generator. The following sections describe how its operation can be configured using the SRGR bit-fields.

TABLE 63.

Sample Rate Generator Register (SRGR) Bit-Field Summary	
Name	Function
GSYNC	Sample rate generator clock synchronization. Only used when the external clock (CLKS) drives the sample rate generator clock (CLKSM=0). GSYNC=0, the sample rate generator clock (CLKG) is always running. GSYNC=1, the sample rate generator clock (CLKG) is synchronized and frame sync signal (FSG) is generated only after detecting the receive frame synchronization signal (FSR). Also, frame period, FPER, is a don't care because the period is dictated by the external frame sync pulse.
CLKSP	CLKS Polarity Clock Edge Select. Only used when the external clock CLKS drives the sample rate generator clock (CLKSM=0). CLKSP=0, falling edge of CLKS generates CLKG and FSG. CLKSP=1, rising edge of CLKS generates CLKG and FSG.
CLKSM	MCSP Sample Rate Generator Clock Mode CLKSM = 0, Sample rate generator clock derived from the CLKS pin. CLKSM = 1 Sample rate generator clock derived from CPU clock.
FSGM	Sample Rate Generator Transmit frame synchronization mode. Used when FSXM=1 in XCR. FSGM=0, Transmit frame sync signal (FSX) due to DXR-to-XSR copy. FSGM=1, Transmit frame sync signal driven by the sample rate generator frame sync signal, FSG.
FPER	Frame Period. This determines when the next frame sync signal should become active.
FWID	Frame Width. Determines the width of the frame sync pulse, FSG, during its active period.
CLKGDV	Sample rate generator clock divider. This value is used as the divide-down number to generate the required sample rate generator clock frequency.

[0317]   Data Clock Generation will now be described. When the receive/ transmit clock mode is set to 1 (CLK(R/X) M=1), the data clocks (CLK(R/X)) are driven by the internal sample rate generator output clock, CLKG. A variety of data bit clocks can be selected independently for the receiver and transmitter. These options include:

1) The input clock to the sample rate generator can be either the CPU clock or a dedicated external clock input (CLKS).
2) The input clock (CPU clock or external clock CLKS) source to the sample rate generator can be divided down by a programmable value (CLKGDV) to drive CLKG.

[0318]   The CLKSM bit in the SRGR selects either the CPU clock (CLKSM=1) or the external clock input (CLKSM=0), CLKS, as the source for the sample rate generator input clock. Any divide periods are divide-downs calculated by the sample rate generator and are timed by this input clock selection.

[0319]   The CLK GDV field determines the sample rate generator data bit clock rate. The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter that is pre-loaded by CLKGDV which contains the divide ratio value. The output of this stage is the data bit clock which is output on sample rate generator output, CLKG, and serves as the input for the second and third divider stages. ·

[0320]   CLKG has a frequency equal to 1/(CLKGDV+1) of sample rate generator input clock. Thus, sample generator input clock frequency is divided by a value from 1-256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high state duration is p+1 cycles and the low state duration is p cycles.

[0321]   Bit Clock Polarity is determined by CLKSP. External clock (CLKS) is selected to drive the sample rate generator clock divider by selecting CLKSM=0. In this case, the CLKSP bit in the SRGR selects the edge of CLKS on which sample rate generator data bit clock (CLKG) and frame sync signal (FSG) are generated. Either the falling edge (CLKSP=0) or the rising edge (CLKSP=1) of the external clock can cause transitions on the data bit-rate clock and FSG.

[0322]   Bit Clock and Frame Synchronization will now be described. When CLKS is selected to drive the sample rate generator (CLKSM=0), GSYNC can be used to configure the timing of CLKG relative to CLKS. GSYNC=1 ensures that the MCSP and the external device it is communicating to, are dividing down the CLKS with the same phase relationship. If GSYNC=0, this feature is disabled and therefore CLKG runs freely and is never resynchronized. If GSYNC=1, an active transition on FSR triggers a re-synchronization of CLKG and generation of FSG. CLKG always begins with a high state after synchronization. Also, FSR is always detected at the same edge of CLKS that generates CLKG. Although an external FSR is provided, FSG can still drive internal receive frame synchronization when GSYNC=1. Note that when GSYNC=1, FPER is a don't care because the frame period is determined by the arrival of the external frame sync pulse.

[0323]   Figure 81 and Figure 82 shows this operation with various polarities of CLKS and FSR. These figures assume: FWID=0, for a FSG one CLKG wide.

Note that FPER is not programmed since it is determined by the arrival of the next external frame sync pulse. The figure shows what happens to CLKG when it is initially in sync and GSYNC=1 as well as not in sync with the frame synchronization and GSYNC=1.

[0324]   When GSYNC=1, the transmitter can operate synchronously with the receiver providing:

1) FSX is programmed to be driven by the sample rate generator frame sync FSG (FSGM=1 in the SRGM and FSXM=1 in the XCR). If the input FSR has the timing so that it can be sampled by the falling edge of CLKG, it can be used instead, by setting FSXM=0 in the XCR and connecting FSR to FSX externally.
2) The sample rate generator clock should drive the transmit and receive bit clock (CLK(R/X)M=1 in the SPCR). Therefore, the CLK(R/X) pin should not be driven by any other driving source.

[0325]   Digital Loop Back Mode, DLB, will now be described. Setting DLB=1 in the SPCR enables digital loop back mode. During DLB mode, the DR, FSR, and CLKR are internally connected to DX, FSX, CLKX respectively, through multiplexers as shown in Figure 78. DLB mode allows testing of serial port code with a single DSP device. Figure 79 shows the multiplexing of receiver control inputs during digital loop back mode.

[0326]   Receive Clock Selection, DLB, CLKRM, will now be described. Table 64 shows how the digital loop back bit (DLB) and the CLKRM bit in the PCR can select the receiver clock. In digital loop back mode (DLB=1), the transmitter clock drives the receiver. CLKRM determines whether the CLKR pin is an input or an output.

TABLE 64.

Receive Clock Selection			
DLB in SPCR	CLKRM in RCR	Source of Receive Clock	CLKR Pin
0	0	CLKR pin acts as an input driven by external clock and inverted as determined by CLKRP before being used.	Input
0	1	Sample Rate Generator Output. Clock (CLKG) drives CLKR.	CLKG inverted as determined by CLKRP before being driven out on CLKR.
1	0	CLKX_int drives CLKR_int as selected and inverted.	High Impedance
1	1	CLKX_int drives CLKR_int as selected and inverted.	Output. CLKR inverted as determined by CLKRP before being driven out.

[0327] Transmit Clock Selection, CLKXM will now be described. Table 65 shows how one of two transmit clock sources can be selected.

TABLE 65.

Transmit Clock Selection		
CLKXM in XCR	Source of Transmit Clock	CLKX Pin
0	External clock drives the CLKX input pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Sample rate generator clock, CLKG, drives transmit clock	Output. CLKG inverted as determined by CLKXP before being driven out on CLKX.

[0328] Frame Sync Signal Generation will now be described. Like data bit clocking, data frame synchronization is also independently programmable for the receiver and transmitter for all data delays. Programming options include:

1) A frame pulse with programmable period between sync pulses, and programmable active width using the sample rate generator register (SRGR).
2) The transmit portion may trigger its own frame sync signal generated by a DXR-to-XSR copy.
3) Both the receive and transmit sections may independently select an external frame synchronization on the FSR and FSX pins, respectively.

[0329] Frame Period is determined by FPER The second divider stage in the sample rate generator is a 12-bit counter which counts the generated data clocks from 0 to 4095. Once the value of the counter matches the programmed frame period (FPER), the frame counter is reset to zero and a FSG is generated. Then the counting process restarts. Thus, the value of FPER+1 determines a frame length from 1 to 4096 data bits. When GSYNC=1, FPER is a don't care value. Figure 83 shows a frame of period 16 CLKG periods (FPER=15 or 00001111b).

[0330] Frame Width is determined by FWID. The FWID field controls the active width of the frame sync pulse. When the counter keeping track of the frame period (FPER) reaches the value programmed in FWID, the frame pulse becomes inactive. Thus, the value of FWID+1 determines a active frame pulse width ranging from 1 to 256 data bit clocks. Figure 43 shows a frame with an active width of 2 CLKG periods (FWID=1).

[0331] Receive Frame Sync Selection is determined by DLB, FSRM, and GSYNC. Table 66 shows how various sources may be selected to provide the receive frame synchronization signal. Note that in digital loop back mode (DLB=1) the transmit frame sync signal is used as the receive frame sync signal and that DR is connected to DX internally.

TABLE 66.

Receive Frame Synchronization Selection				
DLB in SPCR	FSRM in PCR	GSYNC in SRGR	Source of Receive Frame Synchronization	FSR Pin
0	0	X	External frame sync signal drives the FSR input pin. This is then inverted as determined by FSRP before being used as FSR_int.	Input
0	1	0	FSR_int driven by Sample Rate Generator Frame Sync signal (FSG)	Output. FSG inverted as determined by FSRP before being driven out on FSR pin.
0	1	1	FSR_int driven by Sample Rate Generator Frame Sync signal (FSG)	Input. The external frame sync input on FSR is used to synchronize CLKG and generate FSG.
1	0	0	FSX_int drives FSR_int. FSX is selected.	High Impedance.
1	X	1	FSX_int drives FSR_int and is selected.	Input. External FSR not used for frame synchronization but still used to synchronize CLKG and generate FSG since GSYNC=1.
1	1	0	FSX_int drives FSR_int and is selected.	Output. Receive (same as transmit) frame synchroniza-tion inverted as determined by FSRP before being driven out.

[0332] Transmit Frame Sync Signal Selection is determined by FSXM, and FSGM. Table 67 shows how the source of transmit frame synchronization pulses can be selected. The three choices are:

1) External frame sync input.
2) The sample rate generator frame sync signal, FSG.
3) A signal that indicates a DXR-to-XSR copy has been made.

TABLE 67.

Transmit Frame Synchronization Selection			
FSXM in PCR	FSGM in SRGR	Source of Transmit Frame Synchronization	FSX Pin
0	X	External frame sync input on FSX pin. This is inverted by FSXP before being used as FSX_int.	Input.
1	1	Sample Rate Generator Frame Sync signal (FSG) drives FSX_int.	Output. FSG inverted by FSXP before being driven out on FSX pin.
1	0	A DXR-to-XSR copy activates transmit frame sync signal.	Output. One bit clock wide signal inverted by FSXP before being driven out on FSX pin.

[0333] To facilitate detection of frame synchronization, the receive and transmit CPU interrupts (RINT and XINT)

may be programmed to detect frame synchronization by setting RINTM=XINTM=10b, in the SPCR. Unlike other types of serial port interrupts, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In that case, the FS(R/X)M and FS(R/X)P still select the appropriate source and polarity of frame synchronization. Thus even when the serial port is in reset state, these signals are synchronized to CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receive and transmit portions of the serial port. Thus, a new frame synchronization pulse can be detected, after which the CPU can take the serial port out of reset safely.

[0334] Various examples of MCSP configurations will now be described. Figure 84 shows MCSP configuration to be compatible with the Mitel ST-Bus which is a double rate ST-Bus. Note that this operation is running at maximum packet frequency. Various control bits are set as follows:

a) CLK(R/X)M=1, CLK(R/X)_int generated internally by sample rate generator

b) GSYNC=1, synchronize CLKG with external frame sync signal input on FSR. Also note that FSR is regenerated internally to form a minimum pulse width.

c) CLKSM=1, External clock (CLKS) drives the sample rate generator.

d) CLKSP=0, falling edge of CLKS generates CLKG and thus CLK(R/X)_int.

e) CLKGDV=1, receive clock (shown as CLKR) is half of CLKS frequency.

f) FS(R/X)P=1, active-low frame sync pulse.

g) (R/X)FRLEN1= 11111b, 32 words per frame.

h) (R/X)WDLEN1=0, 8-bit word.

i) (RX)PHASE=0, single phase frame and thus (R/X)FRLEN2=X.

j) (R/X)DATDLY=0, no data delay.

[0335] Figure 85 illustrates a single rate ST-BUS. This example is the same as the ST-BUS example except for the following:

a) CLKGDV=0, CLKS drives CLK(R/X)_int without any divide down (single rate clock).

b) CLKSP=1, rising edge of CLKS generates internal clocks CLKG, CLK(R/X)_int.

[0336] The rising edge of CLKS is used to detect the external FSR. This external frame sync pulse is used to re-synchronize internal MCSP clocks and generate frame sync for internal use. Note that the internal frame sync is generated so that it is wide enough to be detected on the falling edge of internal clocks.

[0337] Figure 86 illustrates a Double Rate Clock Example. This example is the same as the ST-BUS example except for the following:

a) GSYNC=0, CLKS drives CLKG. CLKG runs freely and is not re- synchronized by FSR.

b) FS(R/X)M=0, frame synchronization is externally generated. The framing pulse is wide enough to be detected.

c) FS(R/X)P=0, active-high input frame sync signal.

d (R/X)DATDLY=1, data delay of 1-bit.

e) CLKSP=1, rising edge of CLKS generates CLKG and thus CLK(R/X).

[0338] Multi-channel Selection Operation will now be described. Multiple channels can be independently selected for the transmitter and receiver by configuring the MCSP with a single phase frame as described earlier. The number of words per frame represented by (R/X)FRLEN1, denotes the number of channels available for selection.

[0339] Each frame represents a time-division multiplexed (TDM) data stream. In using time-division multiplexed data streams, the CPU may only need to process a few of them. Thus, to save memory and bus bandwidth, multi-channel selection allows independent enabling of particular channels for transmission and reception. Up to 32 channels in an up to 128 channel bit stream can be enabled at any given time, but a dynamic scheme (described later) allows any combination of the 128 channels to be processed.

[0340] If a receive channel is not enabled, the following results:

1) RRDY is not set to 1 upon reception of the last bit of the word.

2) RBR is not copied to DRR upon reception of the last bit of the word. Thus, RRDY is not set active. This feature also implies that no interrupts or synchronization events are generated for this word.

[0341] If a transmit channel is not enabled, the following results:

1) DX is in high impedance.

2) A DXR-to-XSR transfer is not automatically triggered at the end of serial transmission of the related word.

3) XEMPTY- and XRDY similarly are not affected by the end of transmission of the related serial word.

[0342] A transmit channel which is enabled can have its data masked or transmitted. When masked, the DX pin will be forced to high impedance although the transmit channel is enabled.

[0343] The following control registers are used in multi-channel operation:

The Multi-channel Control (MCR) Register

The Transmit Channel Enable (XCER) Register

The Receive Channel Enable (RCER) Register

[0344] The use of these registers in controlling multi-channel operation is described in the following sections.

[0345] The MCR is illustrated in Figure 87 and described in Table 68.

TABLE 68.

Multi-Channel Control Register Bit-Field Descriptions	
Name	Function
RMCM	Receive Multi-channel Selection Enable RMCM = 0, all 128 channels enabled. RMCM = 1, all channels disabled by default. Required channels are selected by enabling RP(A/B)BLK and RCER appropriately.
XMCM	Transmit Multi-channel Selection Enable XMCM = 00b, all channels enabled without masking (DX is always driven and never placed in a high impedance state). XMCM = 01b, all channels disabled and therefore masked by default. Required channels are selected by enabling XP(A/B)BLK and XCER appropriately. Also, these selected channels are not masked and therefore DX is always driven. XMCM = 10b, all channels enabled, but masked. Selected channels enabled via XP(A/B) BLK and XCER are unmasked. XMCM = 11b, all channels disabled and therefore masked by default. Required channels are selected by enabling RP(A/B)BLK and RCER appropriately. Selected channels can be unmasked by RP(A/B)BLK and XCER. This mode is used for symmetric transmit and receive operation.
RCBLK/ XCBLK	Receive/Transmit Current Block. (R/X)CBLK = 000b, Block 0. Channel 0 to channel 15 (R/X)CBLK = 001b, Block 1. Channel 16 to channel 31 (R/X)CBLK = 010b, Block 2. Channel 32 to channel 47 (R/X)CBLK = 011b, Block 3. Channel 48 to channel 63 (R/X)CBLK = 100b, Block 4. Channel 64 to channel 79 (R/X)CBLK = 101b, Block 5. Channel 80 to channel 95 (R/X)CBLK = 110b, Block 6. Channel 96 to channel 111 (R/X)CBLK = 111b, Block 7. Channel 112 to channel 127
RPBBLK / XPBBLK	Receive/Transmit Partition B Block. (R/X)PBBLK= 00b, Block 1. Channel 16 to channel 31 (R/X)PBBLK= 01b, Block 3. Channel 48 to channel 63 (R/X)PBBLK= 10b, Block 5. Channel 80 to channel 95 (R/X)PBBLK= 11b, Block 7. Channel 112 to channel 127
RPABLK / XPABLK	Receive/Transmit Partition A Block. (R/X)PABLK= 00b, Block 0. Channel 0 to channel 15 (R/X)PABLK= 01b, Block 2. Channel 32 to channel 47 (R/X)PABLK= 10b, Block 4. Channel 64 to channel 79 (R/X)PABLK= 11b, Block 6. Channel 96 to channel 111

[0346] Multi-channel mode can be enabled independently for receive and transmit by setting RMCM=1 and XMCM

to a non-zero value in the MCR, respectively.

[0347] A total of 32 out of the available 128 channels may be enabled at any given point in time. The 128 channels comprise eight blocks (0 through 7) and each block has 16 contiguous channels. Further, even-numbered blocks 0, 2, 4, and 6 belong to Partition A, and odd-numbered blocks 1, 3, 5, and 7 belong to Partition B.

[0348] The number of channels enabled can be updated during the course of a frame to allow any arbitrary group of channels to be enabled. This feature is accomplished using an alternating ping-pong scheme controlling two blocks (one odd-numbered and other even-numbered) of 16 contiguous channels each, at any given time within the frame. Thus one block belongs to Partition A and the other to Partition B.

[0349] Any two out of the eight 16-channel blocks may be selected, yielding a total of 32 channels that can be enabled. The blocks are allocated on 16-channel boundaries within the frame as shown in Figure 88. (R/X)PABLK and (R/X) PBBLK fields in the MCR determine the blocks that get selected in partition A and B respectively. This enabling is performed independently for transmit and receive.

[0350] Transmit data masking allows a channel enabled for transmit to have its DX pin set to high impedance state during its transmit period. In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for receive as multiple reception cannot cause serial bus contention.

[0351] The following gives a description of the multi-channel operation during transmit for various XMCM values:

XMCM=OOb: The serial port will transmit data over the DX pin for as many number of words as programmed in XFRLEN1. Thus, DX is driven and never masked.

XMCM=Olb: Required channels or only those words that need to be transmitted are selected via XP(A/B)BLK and XCER.

[0352] Therefore only these selected words will be written to DXR and ultimately transmitted. In other words, if XINTM=OOb which implies that an XINT will be generated for every DXR-to-XSR copy, the number of XINT generated will be equal to the number of channels selected via XCER (and NOT equal to XFRLEN1).

XMCM=10b: For this case, all channels are enabled which means all the words in a data frame (XFRLEN1) will be written to DXR and DXR-to-XSR copy occurs at their respective times. However, DX will be driven only for those channels that are selected via XP(A/B)BLK and XCER and placed in a high impedance state otherwise. In this case, if XINTM=OOb, the number of interrupts generated due to every DXR-to-XSR copy would equal the number of words in that frame (XFRLEN1).

XMCM=11b: This mode is basically a combination of XMCM=01b and 10b cases so that symmetric transmit and receive operation is achieved. All channels are disabled and therefore DR and DX are in high impedance state. For receive, a RBR-to-DRR copy occurs only for those channels that are selected via RP(A/B)BLK and RCER. If RINT were to be generated for every RBR-to-DRR copy, it would occur as many times as the number of channels selected in RCER (and NOT the number of words programmed in RFRLEN1). For transmit, the same block that is used for reception is used in order to maintain symmetry and thus XP(A/B)BLK is a don't care. DXR is loaded and DXR-to-XSR copy occurs for all the channels that are enabled via RP(A/B)BLK . However, DX will be driven only for those channels that are selected via XCER. Note that the channels enabled in XCER can only be a subset or same as those selected in RCER. Therefore, if XINTM=OOb, transmit interrupts to the CPU would be generated as many times as the number of channels selected in RCER (not XCER).

[0353] Figures 89A-D shows the activity on the MCSP pins for all the above modes with the following conditions:

a) (R/X)PHASE=0, single phase frame for multi-channel selection enabled
b) FRLEN1=011b, 4-word frame
c) WDLEN1=any valid serial word length

[0354] Note that in Figures 49A-49D, the arrows showing where the various events occur is only a sample indication. Wherever possible, there is a time window in which these events could occur.

[0355] The Receive Channel Enable (RCER) and Transmit Channel Enable (XCER) registers are used to enable any of the 32 channels for receive and transmit, respectively. Out of the 32 channels, 16 channels belong to a block in partition A and the other 16 belong to a block in partition B. They are shown in Figure 90 and Figure 91, (R/X)CEA and (R/X)CEB register fields shown in Table 69 enable channels within the 16 channel wide blocks in partitions A and B, respectively. The (R/X)PABLK and (R/X)PBBLK fields in the MCR select which 16-channel blocks get selected.

TABLE 69.

Receive/Transmit Channel Enable Register Bit-Field Description	
Name	Function
(R/X)C EAn 0 ≤ n of ≤ 15	Receive/Transmit Channel Enable (R/X)CEA n=0, Disables reception/transmission nth channel in an even-numbered block in partition A (R/X)CEA n=1, Enables reception/transmission of nth channel in an even-numbered block in partition A
(R/X)C EBn 0 ≤ n of ≤ 15	Receive/Transmit Channel Enable (R/X)CEB n=0, Disables reception/transmission nth channel in odd-numbered block in partition B. (R/X)CEB n=1, Enables reception/transmission of nth channel in odd-numbered block in partition B.

[0356] Channel selection may be changed. Using the multi-channel selection feature, a static group of 32 channels may be enabled, and will remain enabled with no CPU intervention until this allocation requires modification. An arbitrary number, group, or all of the words/channels within a frame can be accessed, by updating the block allocation registers during the course of the frame in response to the end-of-block interrupts.

[0357] However, care must be used when changing the selection, not to affect the currently selected block. The currently selected block is readable through the RCBLK and XCBLK fields in the MCR for receive and transmit respectively. The associated channel enable register cannot be modified if it is selected by the appropriate (R/X)P(A/B)BLK register to point toward the current block. Similarly the (R/X)PABLK and (R/X)PBBLK fields in the MCR cannot be modified while pointing to or being changed to point to the currently selected block. Note that if the total number of channels is 16 or less, the current partition is always pointed to. In this case, only a reset of the serial port can change enabling.

[0358] Update Interrupt is provided. At the end of every 16-channel block boundary during multi-channel operation, the receive interrupt (RINT) or transmit interrupt (XINT) to the CPU is generated if RINTM=01b or XINTM=01b in the SPCR, respectively. This interrupt indicates a new partition has been crossed. Then the current partition may be checked and the selection of blocks in the A and/or B partitions may be changed if they do not point to the current block. These interrupts are two CPU clock long active high pulses. Note that if RINTM=XINTM=01b when (R/X)MCM=0 (non-multi-channel operation), it does not generate any interrupts.

[0359] The Clock Stop Mode, CLKSTP, will now be described. The clock stop mode provides compatibility with the SPI™ protocol. Clock stop mode works with single-phase frame configuration. Clock stop mode bit-field (CLKSTP) in the SPCR, allows serial clocks to be stopped between transfers using one of four possible timing variations as shown in Table 70.

[0360] When using SPI-type of interface, frame sync occurs somewhat differently from that in other serial port modes of operation. In these types of interfaces, framing is essentially performed by the presence or absence of clock, and the only other signal involved is the slave enable signal, which simply serves to enable the serial data output driver on the slave device (device not outputting clock). Some devices do not have the slave enable. In order to interface to devices with and without the slave enable, an alternative framing scheme is used. When the MCSP is configured to operate in SPI mode, both the transmitter and the receiver operate together as a master or as a slave.

[0361] When the MCSP is the master device, FSX is set as an output (generated with a load of DXR), and connected to FSR and to the enable input on the slave device, if required. The MCSP generates a continuous clock internally and gates the clock off to the external interface when transfers are complete. In this case receive clock is provided from the internal, continuously running version, so the receiver and transmitter both work internally as if clocks do not stop.

[0362] When the MCSP is the slave device, the internal serial port logic performs transfers using only the exact number of input clock pulses per data bits. If the master device provides a slave enable signal, it is connected to FSX/FSR and is used in its asynchronous form. FSX, then, controls only the initial drive of data to the DX pin. If the master device does not provide a slave enable, the external FSR/FSX pins are connected to an active (as defined by FSX/FSR polarity control bits) level, and data is driven to the DX pin as soon as DXR is loaded. The input clock and frame sync from the master is synchronized to the CPU clock to allow reset. The first data to be transmitted is available on the DX, but is enabled only after detection of SPI clock.

[0363] In clock stop mode, for both the transmitter and receiver, stopping of clocks is handled differently depending on whether the clocks are externally or internally generated. In the case where clocks are externally generated, the external device that is generating clocks holds clocks appropriately between transmitted words. When serial port clock

is internally generated, this clock runs continuously, and is simply gated off to the external interface when the transfer is complete. In this case, transfers are performed in the same fashion as with external free running clocks.

[0364] CLKSTP selects the appropriate clock scheme for a particular SPI interface as shown in Table 70 and Figure 92. CLKSTP bit-field in the SPCR selects one of the following:

1) Whether clock stop mode is enabled or not.
2) In clock stop mode, whether the clock is high or low when stopped.
3) In clock stop mode, whether first clock edge occurs at the start of the first data bit or at the middle of the first data bit.

TABLE 70.

Serial Port Clock Scheme	
CLKSTP	Clock Scheme
000	Clock Stop Mode Disabled
100	Low inactive state without delay: The MCSP transmits data on the rising edge of the CLKX and receive data on the falling edge of the CLKR.
101	Low inactive state with delay: The MCSP transmits data one half cycle ahead of the rising edge of CLKX and receives data on the falling edge of the CLKR.
110	High inactive state without delay: The MCSP transmits data on the falling edge of the CLKX and receive data on the rising edge of the CLKR.
111	High inactive state with delay: The MCSP transmits data one half cycle ahead of the falling edge of CLKX and receives data on the rising edge of the CLKR.

[0365] Two conditions allow the serial port pins (CLKX, FSX, DX, CLKR, FSR, and DR) to be used as general purpose I/O rather than serial port pins:

1) The related portion (transmitter or receiver) of the serial port is in reset; (R/X)RST-=0 in the SPCR, and
2) General purpose I/O is enabled for the related portion of the serial port; (R/X)IOEN=1 in the PCR.

[0366] The Pin Control Register shown in Figure 56 has bits that configure each of the MCSP pins as general purpose inputs or outputs. Table 71 shows how this is achieved. In the case of FS(R/X), FS(R/X)M=0 configures the pin as an input and FS(R/X)M=1 configures that pin as an output. When configured as an output, the value driven on FS(R/X) is the value stored in FS(R/X)P. If configured as an input, the FS(R/X)P becomes a read only bit that reflects the status of that signal. CLK(R/X)M and CLK(R/X)P work similarly for CLK(R/X). When the transmitter is selected as general purpose I/O, the value of the DX_STAT bit in the PCR is driven onto DX. DR is always an input and its value is held in the DR_STAT bit in the PCR. To configure CLKS as a general purpose input, both the transmitter and receiver has to be in reset state and (R/X)IOEN=1, because it is always an input to the MCSP and affects both transmit and receive operations.

TABLE 71.

Configuration of Pins as General Purpose I/O					
Pin	General Purpose I/O Enabled by Setting Both	Selected as Output	Output Value Driven From	Selected as Input	Input Value Readable on
CLKX	XRST-=0 XIOEN=1	CLKXM=1	CLKXP	CLKXM= 0	CLKXP
FSX	XRST-=0 XIOEN=1	FSXM=1	FSXP	FSXM=O	FSXP
DX	XRST-=0 XIOEN=1	always	DX_STAT	never	does not apply

TABLE 71. (continued)

Configuration of Pins as General Purpose I/O					
Pin	General Purpose I/O Enabled by Setting Both	Selected as Output	Output Value Driven From	Selected as Input	Input Value Readable on
CLKR	RRST-=0 RIOEN=1	CLKRM=1	CLKRP	CLKRM= 0	CLKRP
FSR	RRST-=0 RIOEN=1	FSRM=1	FSRP	FSRM=0	FSRP
DR	RRST-=0 RIOEN=1	never	does not apply	always	DR_STA T
CLKS	RRST-=XRST-=0 RIOEN=XIOEN=1	never	does not apply	always	CLKS_S TAT

Timer

Overview

[0367]   The device has two 32-bit general-purpose timers that may be used to:

Time events
Count events
Generate pulses
Interrupt the CPU
Send synchronization events to the DMA

[0368]   The timer has two signaling modes and can be clocked by an internal or an external source. The timer has an I-/O (TIM) that functions as an input clock, as an output clock, or as a general-purpose I-/O pin.

[0369]   With an internal clock, for example, the timer can signal an external A/D converter to start a conversion, or it can trigger the DMA controller to begin a data transfer. With an external clock, for example, the timer can count external events and interrupt the CPU after a specified number of events. Figure 93 shows a block diagram of the timers.

Timer Registers

[0370]   Table 72 describes the three registers that configure timer operation.

TABLE 72.

Timer Registers				
Hex Byte Address		Name	Description	Section
Timer 0	Timer 1			
01940000	01980000	Timer Control	Determines the operating mode of the timer, monitors the timer status, and controls the function of the TIM pin.	9.2.1
01940004	01980004	Timer Period	Contains the number of timer input clock cycles to count. This number controls the TSTAT signal frequency.	9.2.2
01940008	01980008	Timer Counter	Current value of the incrementing counter.	9.2.3

Timer Control Register

[0371]   Figure 94 shows the timer control register. Table 73 describes the bitfields in this register.

TABLE 73.

Timer Control Register Bitfield Description		
Bitfield	Description	Section
FUNC	Function of TIM pin. FUNC=0, TIM is a general-purpose I-/O pin. FUNC=1, TIM is a timer pin.	9.5, 9.6
I-/O	Input Output. Used only when FUNC=0. I-/O=0, TIM is a general-purpose input pin. I-/O=1, TIM is a general-purpose output pin.	9.5, 9.6
DATOUT	Data output. When FUNC=0 and I-/O=1 and CLKSRC=0, Value driven out on TIM. When FUNC=0 and I-/O=1 and CLKSRC=1, Value driven out on TIM and driven to timer input clock.	9.6
DATIN	Data in. Value on TIM pin.	9.6
GO	GO bit. Resets and starts the timer counter. GO=0, no effect to timer. GO=1, and HLD=0, counter register zeroed and begins counting on next clock.	9.3
HLD-	Hold. Counter may be read or written regardless of HLD value. HLD-=0, counter disabled and held in current state. HLD-=1, counter allowed to count.	9.3
C/P-	Clock/pulse mode. C/P-=0, pulse mode, TSTAT active one CPU clock after timer reaches timer period. PWID determines when it goes inactive. C/P-=1, clock mode, TSTAT has 50% duty cycle with high and low periods each one countdown period wide.	9.7
PWID	Pulse Width. Only used in pulse mode (C/P-=0). The TSTAT goes inactive one clock after the timer counter does not equal PWID (0 or 1 as programmed.)	9.7
CLKSRC	Timer Input Clock Source CLKSRC=0, value on TIM pin. CLKSRC=1, CPU clock/4.	9.5

INV	Inverter Control. Only used when FUNC=1.
	INV=0, no effect to timer or TIM pin.
	INV=1,
	    With an external clock source (when CLKSRC=O) and:
	        The value of the TIM pin is inverted before driving the timer input clock.
	    With an internal clock source (when CLKSRC=1) and this drives the TIM pin:
	        The value of TSTAT is inverted before being driven on TIM.
		9.5, 9.6

TSTAT	Timer status. Value of timer output.	9.7

Timer Period Register

[0372]   The timer period register (Figure 95) contains the number of timer input clock cycles to count. This number controls the frequency of TSTAT.

Timer Counter Register

[0373]   The timer counter register (Figure 96) increments whenever enabled to count. Upon reaching the value in the timer period register, it resets to 0 on the next CPU clock. each cycle of the timer input clock.

Resetting the Timer and Enabling Counting: GO and HLD-

[0374]   Table 74 shows how the GO and HLD- enable basic features of timer operation.

5

TABLE 74.

Timer GO and HLD Bitfield Operation			
Operation	GO	HLD-	Description
Holding the Timer	0	0	Counting is disabled.
Restarting the Timer after Hold	0	1	Timer proceeds from state before hold. The timer counter is NOT reset.
Reserved	1	0	undefined
Starting the Timer	1	1	Timer counter resets to 0 and starts counting whenever enabled. Once set, GO self-clears.

10

15

Initializing the Timer

20   [0375]   Configuring a timer requires three basic steps:

1. If the timer is not currently held, place the timer in hold. Note that after device reset, the timer is already in the hold state.
2. Write the desired value to the timer period register.
25   3. Start the timer by setting the GO and HLD bits of the timer control register to 1 and simultaneously writing the desired values to the timer control register.

Timer Counting

30   [0376]   The timer counter runs at the CPU clock rate. However, counting is enabled by on the low-to-high transition from one CPU clock to the next of one of the timer count enable source. This transition is detected by the edge detect circuit shown in Figure 93. Each time an active transition is detected a one CPU clock wide clock enable pulse is generated. To the user, this makes the counter appear as if it were getting clocked by the count enable source. Thus, this count enable source is referred to as the clock source.

35   [0377]   Upon reaching a value equal to the timer period register, the timer is reset to zero. This resetting occurs on the next CPU clock after the timer counter and the timer period match. Thus, the counter counts from zero to N. Consider the case where the period is 2 and the CPU clock/4 was selected as the timer clock source (CLKSRC=1). Once started the timer would count, the following sequence: 0, 0, 0, 0, 1, 1, 1, 1, 2, 0, 0, 0, 1, 1, 1, 1, 2, 0, 0, 0 Note that although the counter counts from 0 to 2, the period is 8 (=2*4) CPU clock cycles rather than 12 (=3*4) CPU clock cycles. Thus,
40   the countdown period is TIMER PERIOD not TIMER PERIOD+1.

Timer Clock Source Selection: CLKSRC

[0378]   Low to high transitions of the timer input clock allow the timer counter to increment. Three sources are available
45   to drive the timer input clock:

1. The input value on the TIM pin. This signal is synchronized to prevent any metastability caused by asynchronous external inputs.
2. The CPU Clock/4.
50   3. The value of the DATOUT.

[0379]   Table 75 shows how the CLKSRC, FUNC, and I-/O bitfields in the timer control register can select which one drives the timer input clock.

55

TABLE 75.

Timer Input Clock Source and TIM Function Selection					
CLKS RC	FUNC	I/O	IN V	Clock Source	TIM Function
0	0	0	X	TIM input.	General purpose input and timer input source.
0	0	1	X	DATOUT Value.	General purpose output driven by DATOUT.
0	1	X	0	TIM input.	Timer input clock source.
0	1	X	1	Inverted TIM input.	Inverted timer input clock source.
1	0	0	X	CPU clock/4	General purpose input.
1	0	1	X	CPU clock/4	General purpose output driven by DATOUT.
1	1	X	0	CPU clock/4	TSTAT driven out.
1	1	X	1	CPU clock/4	Inverted TSTAT driven out.

TIM Pin Function

[0380]   Table 75 also shows the use of the TIM pin. The value of the TIM pin is always captured in the DATIN bit in the timer control register. As shown in Figure 93, the source of DATIN is always passed through a synchronizer to avoid metastability when TIM is an input pin. Also note that INV is only used when TIM is configured as a timer pin (FUNC=1).

Timer Pulse Generation

[0381]   The two basic pulse generation modes are pulse mode and clock mode, as shown in Figure 97 and Figure 98, respectively. One can select the mode with the C/P- bit of the timer global control register. Note that in pulse mode, PWID in the timer control register can set the pulse width to either one or two input clock periods. The purpose of this feature is to provide minimum pulse widths in the case where TSTAT drives the TIM output. TSTAT drives this pin when TIM is used as a timer pin (FUNC=1) and the CPU clock/4 is the clock source (CLKSRC=0). Thus, the TIM pulse width is either 4 or 8 CPU clocks wide. Table 76 details equations for various TSTAT timing parameters in pulse and clock modes.

TABLE 76.

TSTAT Parameters in Pulse and Clock Modes				
Mode	Frequency	Period	Width High	Width Low
Pulse	f (clocksource) timer period register	timer period register f (clocksource)	(PWID +1) f (clocksource)	timer period register - (PWID 1) f (clocksource)

Mode	Frequency	Period	Width High	Width Low
Clock	f (clocksource) 2 * timer period register	2 * timer period register f (clocksource)	timer period register f (clocksource)	timer period register f (clocksource)

Boundary Conditions in the Control Registers

[0382]   Certain boundary conditions affect timer operation:

1. Zero timer period register and counter register. After device reset and before the timer starts counting, TSTAT is held at 0. After the timer starts running by setting GO=1, when the period and counter registers are zero, the operation of the timer depends on the C/P mode selected. In pulse mode, the TSTAT=1 regardless whether or not it is held. In clock mode, when the timer is held (HLD-=0), TSTAT keeps it's previous value and when HLD=1, TSTAT toggles with a frequency of (½ CPU clock frequency).

2. Counter overflow. When the counter register is set to a value greater than the value of the period register, the counter reaches its maximum value (FFFFFFFFh), rolls over to 0, and continues.

3. Writing to registers of an active timer. Writes from the peripheral bus override register updates to the counter register and new status updates to the control register.

4. Small timer period values in pulse mode. Note that small periods in pulse mode can cause TSTAT to remain high. This condition occurs when TIMER PERIOD £ PWID + 1.

Timer Interrupts

[0383]   The TSTAT signal directly drives the CPU interrupt as well as a DMA synchronization event. The frequency of the interrupt is the same as the frequency of the TSTAT. Note that low-to-high transitions on TSTAT always occur when the timer counter equals the period register.

Emulation operation

[0384]   During debug using the emulator, the CPU may be halted on an execute packet boundary for single stepping, benchmarking, profiling, or other debug uses. During an emulation halt, the timer halts when the CPU clock/4 is selected as the clock source (CLKSRC=1). Here, the counter is only enabled to count during those cycles when the CPU is not stalled due to the emulation halt. Thus, counting will be re-enabled during single-step operation. If CLKSRC=O, the timer continues counting as programmed.

Intsel

Overview

[0385]   The peripheral set of microprocessor 1 produces 16 interrupt sources. The CPU however has 12 interrupt available for use. The interrupt selector allows a choice of which 12 of the 16 a system needs to use. The interrupt selector also allows for effective change of the polarity of external interrupt inputs.

Available Interrupt Sources

[0386]   Table 77 lists the available interrupts. Note that this table is similar to the DMA synchronization events described earlier herein except for two differences. One difference is that the MCSP generates separate interrupts and DMA synchronization events. The second difference is that DSPINT has been moved to allow a 4-bit encoding.

TABLE 77.

Available Interrupts		
Interrupt Selection Number	Interrupt Acronym	Interrupt Description
0000b	DSPINT	Host Port Host to DSP Interrupt
0001b	TINT0	Timer 0 Interrupt
0010b	TINT1	Timer 1 Interrupt
0011b	SD_INT	EMIF SDRAM Timer Interrupt
0100b	EXT_INT4	External Interrupt Pin 4
0101b	EXT_INT5	External Interrupt Pin 5
0110b	EXT_INT6	External Interrupt Pin 6
0111b	EXT_INT7	External Interrupt Pin 7
1000b	DMA_INT0	DMA Channel 0 Interrupt
1001b	DMA_INT1	DMA Channel 1 Interrupt
1010b	DMA_INT2	DMA Channel 2 Interrupt
1011b	DMA_INT3	DMA Channel 3 Interrupt
1100b	XINT0	MCSP 0 Transmit Interrupt

TABLE 77. (continued)

Available Interrupts		
Interrupt Selection Number	Interrupt Acronym	Interrupt Description
1101b	RINT0	MCSP 0 Receive Interrupt
1110b	XINT1	MCSP 1 Transmit Interrupt
1111b	RINT1	MCSP 1 Receive Interrupt

External Interrupt Signal Timing

[0387] INT4-INT7 and NMI are dedicated external interrupt sources. In addition, the FSR and FSX can be programmed to directly drive the RINT and XINT signals. Because these signals are asynchronous, they are passed through two registers before being sent to either the DMA or CPU. Figure 99 shows the timing of external interrupt signals using INT4 as an example. This diagram is similar to the one in the CPU Reference Guide. However, this diagram also shows the delays for the external interrupt through the two synchronization flip-flops. Note, that this delay is two CPU clock (CLKOUT1) cycles. However, if the INT4 input transitions during the setup and hold time with respect to the CLKOUT1 rising edge, this delay could be as long as 3 CLKOUT1 cycles. Once synchronized, an additional 3 CLKOUT1 cycle delay occurs before the related interrupt flag (IF4) is set.

[0388] The interrupt can only be schedule to be taken one CLKOUT1 cycle later at the earliest as indicated by the active internal interrupt acknowledge (IACK) signal. The interrupt can be postponed or inhibited if not properly enabled. In that case, IACK will be also be postponed. Along with IACK, the CPU sets the INUM signal to indicate which interrupt was taken. Externally, the IACK pin pulse is extended to two CLKOUT2 cycles wide and synchronized to CLKOUT2. Also, the INUM pin signal frames this external IACK with one CLKOUT2 cycle of setup and hold, for a width of 4 CLKOUT2 cycles. Note that even though INUM and IACK in the diagram are not valid on a CLKOUT2 rising edge, the internal circuitry still catches the transition and produces the desired waveforms on the IACK and INUM pins.

Interrupt

[0389] Table 78 shows the interrupt selector registers. The Interrupt Multiplexor Registers determine the mapping between the interrupt sources in Table 77 and the CPU interrupts 4 through 15 (INT4-INT15). The External Interrupt Polarity register sets the polarity of external interrupts.

TABLE 78.

Interrupt Selector Registers		
Hex Byte Address	Name	Description
019C0000	Interrupt Multiplexor High	Selects which interrupts drive CPU interrupts 10-15 (INT10-15)
019C0004	Interrupt Multiplexor Low	Selects which interrupts drive CPU interrupts 4-9 (INT4-INT9)
019C0008	External Interrupt Polarity	Sets the polarity of the external interrupts (EXT_INT4-EXT_INT7)

External Interrupt Polarity Register

[0390] The external interrupt polarity register (Figure 100) allows for a change in the polarity of the four external interrupts (EXT_INT4-EXT_INT7). Normally, a low-to-high transition on an interrupt source is recognized as an interrupt. By setting the related XIP bit in this register to 1, the external interrupt source can be inverted and effectively have the CPU detect high-to-low transitions of the external interrupt. If the related XIP is cleared to 0, then the non-inverted external interrupt is passed and the CPU recognizes a low-to-high transition as signaling an interrupt.

Interrupt Multiplexor Registers

[0391] The INTSEL fields in the Interrupt Multiplexor Registers (Figures 101 and 102) allow mapping the interrupt sources in to particular interrupts. The INTSEL4-INTSEL15 correspond to CPU interrupts INT4-INT15 as shown in Table 79. By setting the INTSEL fields to the value of the desired interrupt selection number in Table 78, any interrupt source may be mapped to any CPU interrupt. Table 79 also shows the default mapping of interrupt sources to CPU

interrupts.

TABLE 79.

Default Interrupt Mapping				
CPU Interrupt	Related INTSEL field	INTSEL Reset Value	Interrupt Acronym	Interrupt Description
INT4	INTSEL4	0100	EXT_INT4	External Interrupt Pin 4
INT5	INTSEL5	0101	EXT_INT5	External Interrupt Pin 5
INT6	INTSEL6	0110	EXT_INT6	External Interrupt Pin 6
INT7	INTSEL7	0111	EXT_INT7	External Interrupt Pin 7
INT8	INTSEL8	1000	DMA_INT0	DMA Channel 0 Interrupt
INT9	INTSEL9	1001	DMA_INT1	DMA Channel 1 Interrupt
INT10	INTSEL10	0011	SD_INT	EMIF SDRAM Timer Interrupt
INT11	INTSEL11	1010	DMA_INT2	DMA Channel 2 Interrupt
INT12	INTSEL12	1011	DMA_INT3	DMA Channel 3 Interrupt
INT13	INTSEL13	0000	DSPINT	Host Port Host to DSP Interrupt
INT14	INTSEL14	0001	TINT0	Timer 0 Interrupt
INT15	INTSEL15	0010	TINT1	Timer 1 Interrupt

Configuring the Interrupt Selector.

[0392]   The interrupt selector registers is meant to be configured once after reset during initialization before enabling interrupts. Once the registers have been set, the interrupt flag register should be cleared by the user after some delay to remove any spurious transitions caused by the configuration. The interrupt selector may be reconfigured during other times, but spurious interrupt conditions maybe detected by the CPU on the interrupts affected by the modified fields.

Power Down Modes and Emulation

[0393]   Since the interrupt selector contains no state machine, there is no unusual action taken during power down or an emulation halt. Register values are preserved in either case.

Clock

Overview

[0394]   An external oscillator drives the on-chip PLL (Phase-Locked Loop) circuit that generates all internal and external clocks. The PLL multiplies the external oscillator frequency by 4 and feeds the resulting clock to CLKOUT1 output pin. The internal version of CLKOUT1 is used by the processor as an instruction cycle clock. Most timing parameters of this device are defined relative to the CLKOUT1 clock and specifically to it's rising edge. CLKOUT2 is another output clock derived in EMIF from CLKOUT1 at half of it's frequency. Two other clock signals are derived from the PLL outputs - SSCLK and SDCLK (also in EMIF). They are used to clock the external SBSRAM and SDRAM synchronous memories.

[0395]   In addition x4 mode, the clock circuit can operate in multiply by 1 mode, where the input clock frequency is the same as the CLKOUT1 output clock frequency. The factors to consider in choosing the multiply factor include board level noise and clock jitter. The x4 mode will minimize the board noise, while the xl mode will reduce the internal clock jitter. The clock mode is controlled by two CLKMODE pins as shown in the following Figure 103.

[0396]   The amount of time that the PLL needs to synchronize to the output frequency depends on the CLKIN and CLKOUT1 frequencies and is typically in the range of tens of microseconds. The synchronization time affects the duration of the Reset signal in that the reset has to be asserted long enough for the PLL to synchronize to the proper output frequency.

[0397] Three PLLFREQ pins identify the range of CLKOUT1 frequencies that the PLL is expected to synchronize to. The PLL also requires 2 bypass capacitors (between PLLV and PLLG), external low-pass filter components (R1, C1 , C2) and an EMI filter. The values for R1, C1, C2 and the filter depend on the CLKIN and CLKOUT1 frequencies.

Pdown

Overview

[0398] Most of the operating power of CMOS logic is dissipated during circuit switching from one logic state to another. By preventing some or all of chip's logic from switching, significant power savings can be realized without losing any data or operational context.

[0399] Power-down mode Pd1 blocks the internal clock inputs at the boundary of the CPU, preventing most of its logic from switching. Pdl effectively shuts down the CPU. Additional power savings are accomplished in power-down mode Pd2, where the entire on-chip clock structure (including multiple buffers) is "halted" at the output of the PLL (see Figure 104).

[0400] PD3 shuts down the entire internal clock tree (like PD2) and also disconnects the external clock source (CLKIN) from reaching the PLL. Wake-up from Pd3 takes longer then wake-up from Pd2 because the PLL needs to be re-locked, just as it does following power-up.

[0401] Pd2 and Pd3 halt the entire chip in both PLL clock modes - x1 and x4. Both the Pd2 and Pd3 signals also assert the PD_ pin for external recognition of these two power-down modes. In addition to power-down modes described in this chapter, the IDLE instruction provides lower CPU power consumption by executing multiple NOPs. The IDLE instruction terminates only upon servicing an interrupt.

[0402] The power-down modes and their wake-up methods are programmed by setting bits 10-14 in the of the Control Status Register (CSR PWRD field) (See Figure 105). Pd2 and Pd3 modes can only be aborted by device Reset, while Pdl mode can also be terminated by an enabled interrupt, or any interrupt (enabled or not), as directed by bits 13 and 14 of the CSR (interrupts are edge driven). When writing to CSR, all bits of the PWRD field should be set at the same time. Logic 0 should be used when writing to reserved fields (bit 15 of CSR). See Table 80

TABLE 80.

Power-Down Mode and Wake-up Selection					
CSR bit 14	CSR bit 13	CSR bit 12	CSR bit 11	CSR bit 10	power-down mode/wake-up method
0	0	0	0	0	no power-down
0	1	0	0	1	Pdl/wake by an enabled interrupt [1]
1	0	0	0	1	Pdl/wake by an enabled or non-enabled interrupt
1	1	0	1	0	Pd2
1	1	1	0	0	Pd3
all other combinations of CSR bits 14-10					reserved

[1] In addition to individual interrupt enables, CSR register GIE and IER register NMIE bits must also be set before an interrupt can be serviced.

Triggering, Wake-up and Effects

[0403] As noted in Table 81, power-down mode pdl takes effect 3-4 instructions after the instruction that caused the power-down (by setting the idle bits in the CSR). See the instruction flow below :

INSTR1 (MVK)	power-down mode is set by this instruction
INSTR2	CPU notifies power-down logic to initiate power-down
INSTR3	power-down signal is sent to CPU and/or peripherals
INSTR4	CPU receives the internal power-down signal
INSTR5	CPU suspends execution before INSTR5 or INSTR6
INSTR6	CPU resumes execution with INSTR5 or INSTR6
INSTR7	normal program execution resumed here

[0404]   The wake-up from Pdl can be triggered by either an enabled interrupt, or any interrupt (enabled or not). The first case is selected by writing a logic 1 to bit 13 of the Control Status Register (PWRD field), and the second case is selected by writing a logic 1 into bit 14 of CSR. If Pdl mode is terminated by a non-enabled interrupt, the program execution returns to the 4th or 5th instruction following the one that caused the power-down (by setting the idle bits in the CSR). Wake-up by an enabled interrupt executes the corresponding Interrupt Service Fetch Packet first, prior to returning to the 4th or 5th instruction following the one that caused the power-down (CSR register GIE and IER register NMIE bits must also be set in order for the ISFP to execute).

TABLE 81.

Characteristics of the Power-Down Modes			
Power Down Mode	Trigger Action	Wake-up Method	Effect
Pd1	Write logic 01001b or 1000b to , bits 14-10 of the CSR	internal interrupt, external interrupt or Reset	CPU halted (except for the interrupt logic)
Pd2	Write logic 11010b to bits 14-10 of the CSR	Reset only	Output clock from PLL is halted, stopping the internal clock structure from switching and resulting in the entire chip being halted. Signal terminal PD_ is driven low. All register and internal RAM contents are preserved. All signal terminals behave the same way as during Reset.
Pd3	Write logic 11100b to bits 14-10 of the CSR	Reset only	Input clock to the PLL is halted, shutting down the PLL and stopping the internal clock structure from switching and resulting in the entire chip being halted. Signal terminal PD_ is driven low. All register and internal RAM contents are preserved. All signal terminals behave the same way as during Reset. Following reset, the PLL needs time to re-lock, just as it does following power-up.

[0405]   Fabrication of data processing device 1 involves multiple steps of implanting various amounts of impurities into a semiconductor substrate and diffusing the impurities to selected depths within the substrate to form transistor devices. Masks are formed to control the placement of the impurities. Multiple layers of conductive material and insulative material are deposited and etched to interconnect the various devices. These steps are performed in a clean room environment.

[0406]   A significant portion of the cost of producing the data processing device involves testing. While in wafer form, individual devices are biased to an operational state and probe tested for basic operational functionality. The wafer is then separated into individual dice which may be sold as bare die or packaged. After packaging, finished parts are biased into an operational state and tested for operational functionality.

[0407]   An alternative embodiment of the novel aspects of the present invention may include other functional circuitries which are combined with the functional circuitries disclosed herein in order to reduce the total gate count of the combined functions. Since those skilled in the art are aware of techniques for gate minimization, the details of such an embodiment will not be described herein.

[0408]   As used herein, the terms "applied," "connected," and "connection" mean electrically connected, including where additional elements may be in the electrical connection path.

[0409]   While the invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various other embodiments of the invention will be apparent to persons

skilled in the art upon reference to this description.

5

10

15

20

25

30

35

40

45

50

55

Appendix A

LIST OF ACRONYMS

AC97	Audio Codec '97 (component specification)
CLKG	Output clock of Sample Rate Generator
CLKGDV	Divider for Sample Rate Generator Clock
CLKR(P/M)	Clock for Receive (Polarity/Mode)
CLKSTP	Clock Stop
CLKX(P/M)	Clock for Transmit (Polarity/Mode)
DLB	Digital Loop Back
DR	Data Receive
DRR	Data Receive Register
DX	Data Transmit
DXR	Data Transmit Register
FPER	Frame Period
FSG	Frame Sync output from sample rate generator
FSGM	FSG Mode
FSR(P/M)	Frame Synchronization (Polarity/Mode) for Receive
FSX(P/M)	Frame Synchronization (Polarity/Mode) for Transmit
FWID	Frame Width
GSYNC	Sample Rate Generator Clock Synchronization
IIS	Inter-IC Sound bus (serial link for digital audio)
IOM	ISDN-Oriented Modular (Architecture and Interfaces)
IOM2	Extended IOM
MCR	Multi-channel Control Register
MCSP	Multi Channel Serial Port
MVIP	Multi-Vendor Integration Protocol
PCR	Pin Control Register

RCBLK	Receive Current Block
RCER	Receive Channel Enable Register
RCR	Receive Control Register
RDATDLY	Receive Data Delay
REVT	Receive Synchronization Event to DMA
RFIG	Receive Frame Ignore
RFRLEN(1/2)	Receive Frame Length for Frame Phase 1 or 2
RFULL	RSR (Receive Shift Register) Full
RINT	Receive Interrupt to CPU
RIOEN	Receive I/O Enable
RJUST	Receive data (in RSR) Justification
RMCM	Receive Multi-Channel Mode
RP(A/B)BLK	Receive Partition A/B Block
RPHASE	Receive (number of) Phases
RRDY	Receiver Ready
RRST	Receiver Reset
RSYNCERR	Receive Synchronization Error
RWDLEN(1/2)	Receive Word Length for Frame Phase 1 or 2
SPCR	Serial Port Control Register
SPI	Serial Peripheral Interface (Synchronous)
SRGR	Sample Rate Generator Register
ST-bus	Serial Telecom Bus (Mitel Semiconductor)
XCBLK	Transmit Current Block
XCER	Transmit Channel Enable Register
XCR	Transmit Control Register
XDATDLY	Transmit Data Delay
XEMPTY	XSR (Transmit Shift Register) Empty
XEVT	Transmit Synchronization Event to DMA
XFIG	Transmit Frame Ignore
XFRLEN(1/2)	Transmit Frame Length for Frame Phase 1 or 2
XINT	Transmit Interrupt to CPU

XIOEN	Transmit I/O Enable
XMCM	Transmit Multi-Channel Mode
XP(A/B)BLK	Transmit Partition A/B Block
XPHASE	Transmit (number of) Phases
XRDY	Transmitter Ready
XRST	Transmitter Reset
XSR	Transmit Shift Register
XWDLEN(1/2)	Transmit Word Length for Frame Phase 1 or 2

Claims

1.-A data processing device, comprising:

a central processing unit (CPU) for executing instructions;
a memory circuit connected to the central processing unit for storing the instructions which are executed by the CPU; and
serial port interface circuitry connected to the CPU, for transmitting and receiving data with dual phase frames, wherein each phase has a different set of parameters.

2. The data processing device of Claim 1, wherein the serial port interface circuitry is further operable for selecting a different number of words for each phase.

3. The data processing device of Claim 2, wherein the serial port interface circuitry further comprises:

a data transmit register connected to a data transmit output, and for receiving a transmit data word having a plurality of bits from the CPU;
frame sync generation circuitry and control circuitry, for causing a serial data stream to be transmitted from the data transmit register via the data transmit port in a time-division multiplexed manner, such that a plurality of data channels is available; and
a multi-channel control register for selecting a portion of the plurality of data channels in which to transmit the transmit data word, such that each bit of the transmit data word is assigned to a selected channel from the plurality of data channels.

4. The data processing device of Claim 3, wherein the serial port interface circuitry further comprises:
a transmit channel enable register connected to the data transmit port, and for disabling transmission of one or more bits of the transmit data word.

5. The data processing device of Claim 4, wherein the serial port interface circuitry further comprises:

a sample rate clock generation circuit connected to a data receive port;
an external clock input connected to the sample rate generation circuit for receiving an external clock signal;
a frame sync signal input connected to the sample rate clock generation circuit for receiving an external frame sync signal;
a sample rate generation register connected to the sample rate generation circuit for controlling the sample rate generation circuit in response to another selected instruction executed by the CPU, the sample rate generation circuit comprising a frame pulse detection circuit connected to the frame sync signal input; and
wherein the sample rate generation circuit is for resynchronizing the sample rate clock generation circuitry to the external frame sync signal in response to the frame pulse detection circuit when a control bit in the sample rate generation register is in a first state, and the sample rate generation circuit is further operable to free-run

when the control bit in the sample rate generation register is in a second state.

5. The data processing device of Claim 5, wherein the sample rate generation circuit further comprises:

a clock divider circuit connected to the external clock input and having a data bit clock output, and for converting the external clock signal having a first frequency to a data bit clock having a second frequency in response to another selected instruction executed by the CPU.

6. The data processing device of Claim 6, wherein the sample rate generation circuit further comprises:

an internal clock input connected to a first input of a multiplexer circuit;
wherein the external clock input is connected to a second input of the multiplexor, and an output of the multiplexor is connected to clock divider circuit; and
wherein the multiplexor is responsive to another selected instruction executed by the CPU.

8. A data processing device, comprising:

a central processing unit (CPU) for executing instructions,
a memory controller connected to said central processing unit, and
a memory circuit connected to said memory controller,

9. The data processing device of Claim 8, further comprising:

a direct memory access (DMA) controller having programmable read and write address circuitry connected to said memory controller.

10. The data processing device of Claim 8 or Claim 9, further comprising:

an external memory interface circuit connected to said memory controller.

11. The data processing device of claim 2, further comprising:

a second memory controller connected to said central processing unit; and
a second memory connected to said second memory controller,

12. The data processing device of any of any of Claims 9 to 11, further comprising:

a host port interface connected to said direct memory access controller.

13. The data processing device of any of Claims 8 to 12, further comprising:

a peripheral bus controller connected to said memory controller, and an external memory interface controller connected to said peripheral bus controller.

14. The data processing device of Claim 13, wherein said external memory interface circuit is arranged to programmably interface with a plurality of different memory types,

15. The data processing device of any of Claims 8 to 14, further comprising:

means for providing at least one external signal for defining the type of memory located at a reset address for said central processing unit,

16. The data processing device of any of Claims 8 to 15, further comprising:

at least one external signal for identifying the size and location of said memory circuit,

17. The data processing device of any of Claims 13 to 16, further comprising:

a programmable timer connected to said peripheral bus controller.

18. The data processing device of any of Claims 8 to 17, further comprising:

a programmable phase-locked loop circuit for providing clock signals to said central processing unit.

19. The data processing device of any of Claims 8 to 18, further comprising:

a power down circuit for programmably preventing clocking signals from reaching preselected portions of said device.

20. A data processing device, comprising:

a central processing unit for executing software instructions stored in a program memory circuit connected to the central processing unit;
a memory circuit for storing data to be processed by the processing device; and
a direct memory access (DMA) controller having read address circuitry and write address circuitry, and for transfering data from or to the memory circuit, the DMA controller further having DMA interrupt circuitry for interrupting the central processing unit.

21. A data processing device, comprising:

a central processing unit (CPU) for executing software instructions stored in a program memory circuit connected to the central processing unit;
a data memory circuit for storing data to be processed by the processing device;
a direct memory access (DMA) controller having read address circuitry and write address circuitry, for transfering data from or to the data memory circuit, the DMA controller further having DMA interrupt circuitry for interrupting the central processing unit;
a peripheral device having address generation circuitry, the address generation circuitry for providing an address for transferring data to or from the data memory circuit; and
auxiliary channel control circuitry for transfering a first data word to the data memory circuit from the peripheral device using the address generation circuitry of the peripheral device and to interrupt the central processor using the DMA interrupt circuitry of the DMA controller.

22. A data processing device, comprising:

a central processing unit (CPU) for executing software instructions stored in a program memory circuit connected to the central processing unit;
a data memory circuit for storing data to be processed by the processing device;
a direct memory access (DMA) controller having programmable read address circuitry and programmable write address circuitry, for transfering data from or to the data memory circuit;
wherein the programmable read address circuitry is arranged to form a subsequent address by incrementing or decrementing a first address by a programmable amount.

23. The data processing device of Claim 22, wherein the programmable amount has a first value for transfers within a frame and a second value for the last transfer of a frame.

24. The data processing device of Claim 22 or Claim 23, further comprising:

circuitry for holding a fixed split destination address, operable to be loaded by a selected instruction executed by the CPU;
circuitry for holding a fixed split source address, operable to be loaded by a selected instruction executed by the CPU;
circuitry for performing a split channel data transfer, for transferring a first data stream by reading a first plurality of data words in response to incrementing or decrementing the read address circuitry and by writing the first plurality of data words to the same split destination address; and
wherein the circuitry for performing a split channel data transfer is further arranged to coincidentally transfer a second data stream by reading a second plurality of data words from the same fixed split source address and by writing the second plurality of data words in response to incrementing or decrementing the write address circuitry.

25. The data processing device of any of Claims 22 to 24, further comprising a FIFO buffer connected to a data bus for receiving a plurality of data words received in response to the read address circuitry, the FIFO buffer operable to hold a portion of the plurality of data words until the portion of the plurality of data words is written in response to the write address circuitry.

26. The data processing device of Claim 25, further comprising a plurality DMA channels, wherein each DMA channel has associated programmable read address circuitry and programmable write address circuitry, each DMA channel operable to transfer data from or to the data memory circuit; and

wherein the FIFO buffer can be selectively associated with any one of the plurality of DMA channels

27. A data processing device, comprising:

a central processing unit (CPU) for to executing software instructions stored in a program memory circuit connected to the central processing unit;
a data memory circuit for to storing data to be processed by the processing device;
a direct memory access (DMA) controller having programmable read address circuitry and programmable write address circuitry, for transfering a frame of data from or to the data memory circuit, wherein a frame of data comprises a plurality of data words transferred in a successive fashion to or from the data memory circuit;
circuitry for selecting a frame synchronization event connected to the DMA controller, operable to be loaded by a selected instruction executed by the CPU; and
wherein the DMA control circuitry is further operable to initiate transfer of a first frame of data only after the selected frame synchronization event occurs.

28. A data processing device, comprising:

a central processing unit (CPU) for executing software instructions stored in a program memory circuit connected to the central processing unit;
a data memory circuit for storing data to be processed by the processing device;
a direct memory access (DMA) controller having programmable read address circuitry and programmable write address circuitry, operable to transfer data from or to the data memory circuit; and

wherein the DMA controller further comprises:

a plurality of control registers, comprising a source address register connected to the read address circuitry and a destination address register connected to the write address circuitry, operable to be loaded by selected instructions execuzed by the CPU;
a plurality of reload registers associated respectively to a portion of the plurality of control registers, operable to be loaded with initialization data by selected instructions executed by the CPU; and
auto-initialization circuitry operable to initialize the control registers by transferring the initialization data from to plurality of reload registers to the respectively connected portion of the plurality of control registers.

29. The data processing device of Claim 28, further comprising:

a plurality of global control registers connected to the DMA controller circuitry, operable to be loaded with control data by selected instructions executed by the CPU; and
wherein a same one of the plurality of global control registers can be selected as source reload register and be associated with the source address register during a first DMA transfer operation, and can thereafter be selected as a destination reload register and be associated with the destination address register during a second DMA transfer operation.

30. A data processing device, comprising:

a central processing unit (CPU) for executing software instructions stored in a program memory circuit connected to the central processing unit;
a data memory circuit for storing data to be processed by the processing device;
a direct memory access (DMA) controller having programmable read address circuitry and programmable write address circuitry, operable to transfer data from or to the data memory circuit;
an output status pin connected to the DMA controller circuitry, operable to indicate DMA controller circuitry status to an external device;
a control register connected to the DMA controller circuitry, operable to be loaded with status pin control data by selected instructions executed by the CPU; and
wherein the DMA control circuitry is operable to provide a first status signal to the output status pin selected from a plurality of status signals in response to the status pin control data.

31. A data processing device, comprising:

a central processing unit (CPU) for executing software instructions stored in a program memory circuit connected to the central processing unit;

a data memory circuit for storing data to be processed by the processing device;

a direct memory access (DMA) controller having read address circuitry and write address circuitry, operable to transfer a frame of data from or to the data memory circuit, wherein a frame of data comprises a plurality of data words transferred in a successive fashion to or from the data memory circuit, the DMA controller further having DMA interrupt circuitry operable to interrupt the central processing unit;

wherein the programmable read address circuitry is operable to form a subsequent address by incrementing or decrementing a first address by a programmable amount, wherein the programmable amount has a first value for transfers within a frame and a second value for the last transfer of a frame;

circuitry for holding a fixed split destination address, operable to be loaded by a selected instruction executed by the CPU;

circuitry for holding a fixed split source address, operable to be loaded by a selected instruction executed by the CPU;

circuitry for performing a split channel data transfer, operable to transfer a first data stream by reading a first plurality of data words in response to incrementing or decrementing the read address circuitry and by writing the first plurality of data words to the same split destination address;

wherein the circuitry for performing a split channel data transfer is further operable to coincidentally transfer a second data stream by reading a second plurality of data words from the same fixed split source address and by writing the second plurality of data words in response to incrementing or decrementing tne write address circuitry;

a FIFO buffer connected to a data bus to receive a plurality of data words received in response to the read address circuitry, the FIFO buffer operable to hold a portion of the plurality of data words until the portion of the plurality of data words is written in response to the write address circuitry;

circuitry for selecting a frame synchronization event connected to the DMA controller, operable to be loaded by a selected instruction executed by the CPU, wherein the DMA control circuitry is further operable to initiate transfer of a first frame of data only after the selected frame synchronization event occurs;

a plurality of control registers, comprising a source address register connected to the read address circuitry and a destination address register connected to the write address circuitry, operable to be loaded by selected instructions executed by the CPU;

a plurality of reload registers associated respectively to a portion of the plurality of control registers, operable to be loaded with initialization data by selected instructions executed by the CPU;

auto-initialization circuitry operable to initialize the control registers by transferring the initialization data from the plurality of reload registers to the respectively connected portion of the plurality of control registers;

an output status pin connected to the DMA controller circuitry, operable to indicate DMA controller circuitry status to an external device;

wherein one of the plurality of control registers connected to the DMA controller circuitry is operable to be loaded with status pin control data by selected instructions executed by the CPU; wherein the DMA control circuitry is operable to provide a first status signal to the output status pin selected from a plurality of status signals in response to the status pin control data;

a peripheral device having address generation circuitry, the address generation circuitry operable to provide an address for transferring data to or from the data memory circuit; and

auxiliary channel control circuitry operable to transfer a first data word to the data memory circuit from the peripheral device using the address generation circuitry of the peripheral device and to interrupt the central processor using the DMA interrupt circuitry of the DMA controller.

32. A data processing device, comprising:

a central processing unit (CPU) for executing instructions, the CPU having a first number of interrupt terminals;

a memory circuit connected to the CPU for storing a plurality of instructions which are executed by the CPU; and

a plurality of interrupt request inputs, wherein the number of interrupt request inputs is greater than the first number of interrupt terminals; and

an interrupt selector connected to the plurality of interrupt request inputs to receive a plurality of interrupt request signals, and having a plurality of outputs connected to the interrupt terminals, the interrupt selector operable to select a subset of the plurality of interrupt request signals and further operable to connect the selected subset to the interrupt terminals.

33. The data processing device of Claim 32, further comprising:

an interrupt multiplexor register connected to the interrupt selector, operable to receive a mapping word having a plurality of map fields from the CPU in response to a selected instruction executed by the CPU, wherein each of the plurality of map fields is associated with a different one of the interrupt terminals; and

wherein the interrupt selector is operable to map a selected interrupt request signal from the plurality of interrupt request signals selected by each of the plurality of map fields to the associated different one of the interrupt terminals.

34. The data processing device of Claim 32 or Claim 33, further comprising:

an interrupt polarity register connected to the interrupt selector, operable to receive a polarity word having a plurality of polarity bits from the CPU in response to a selected instruction executed by the CPU, wherein each of the plurality of polarity bits is associated with a different one of the plurality of interrupt request inputs; and

rein the interrupt selector is further operable to select an active polarity of each of a portion of the plurality of interrupt request signals in response to an associated one of the plurality of polarity bits.
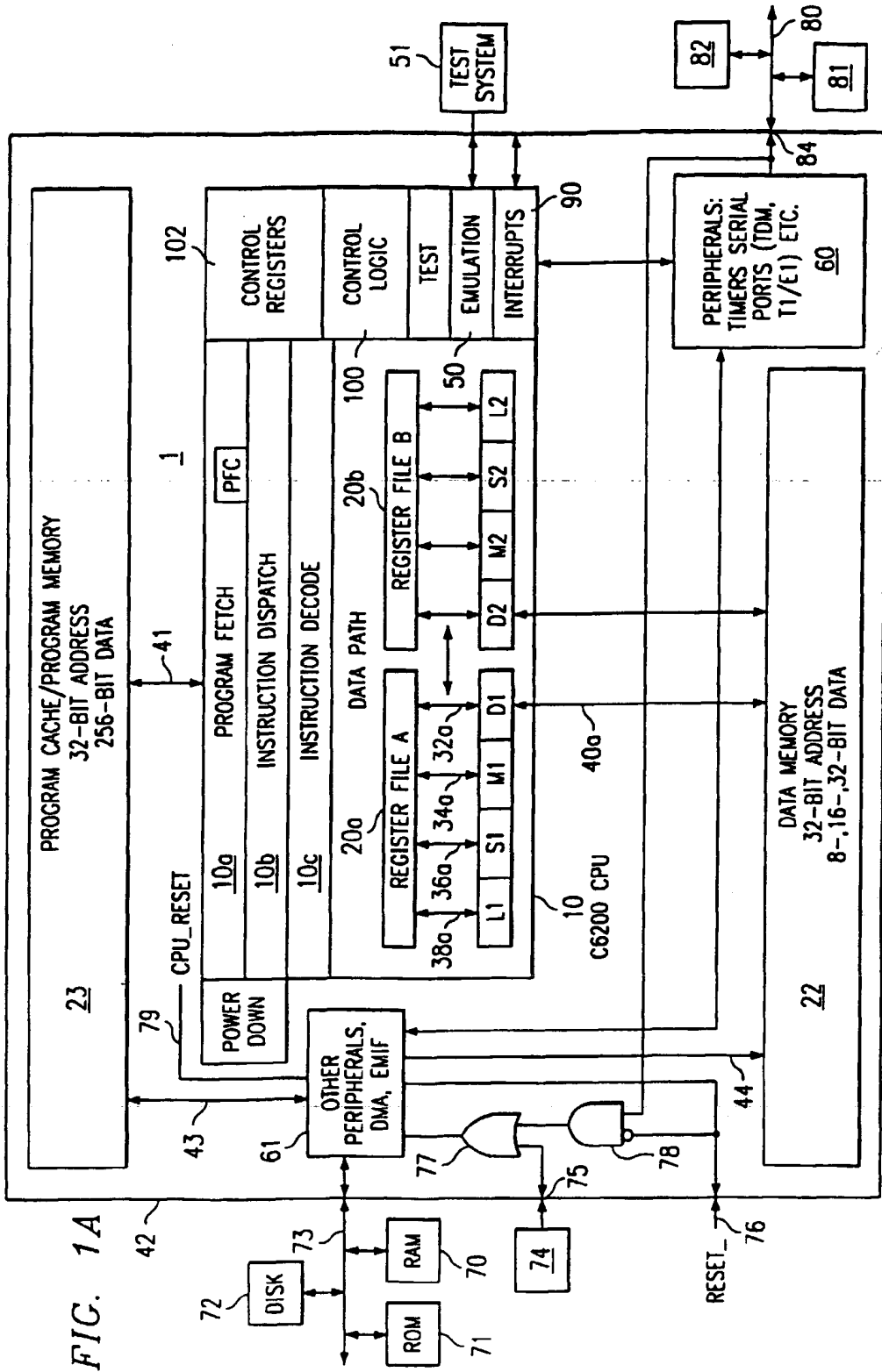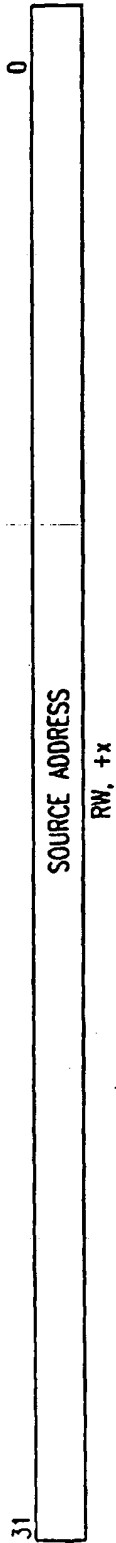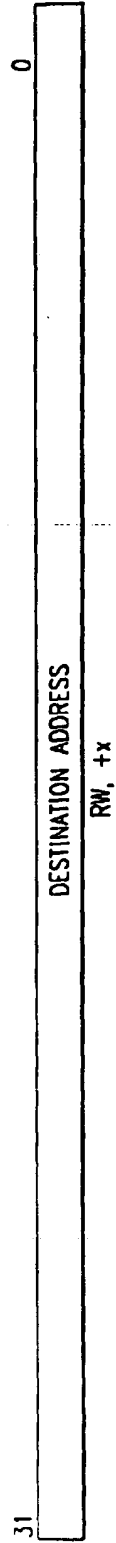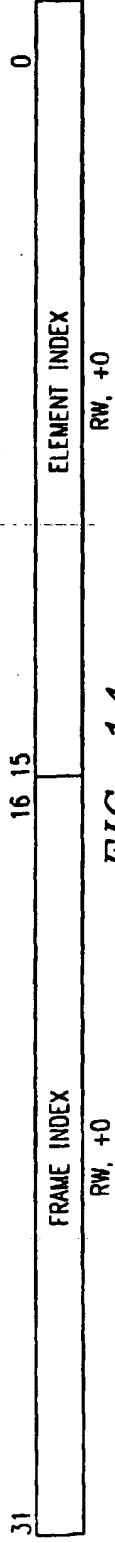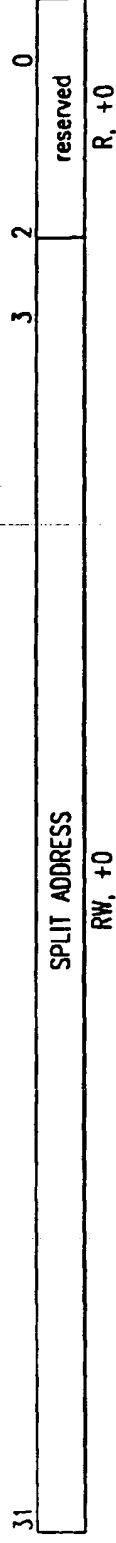
FIG. 1A
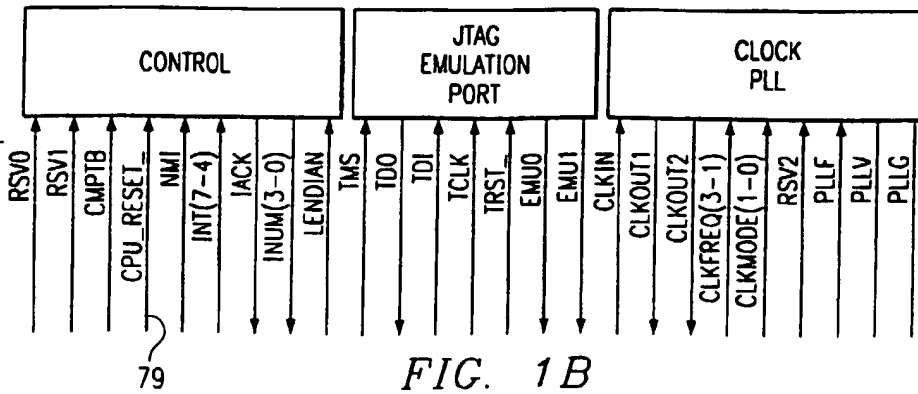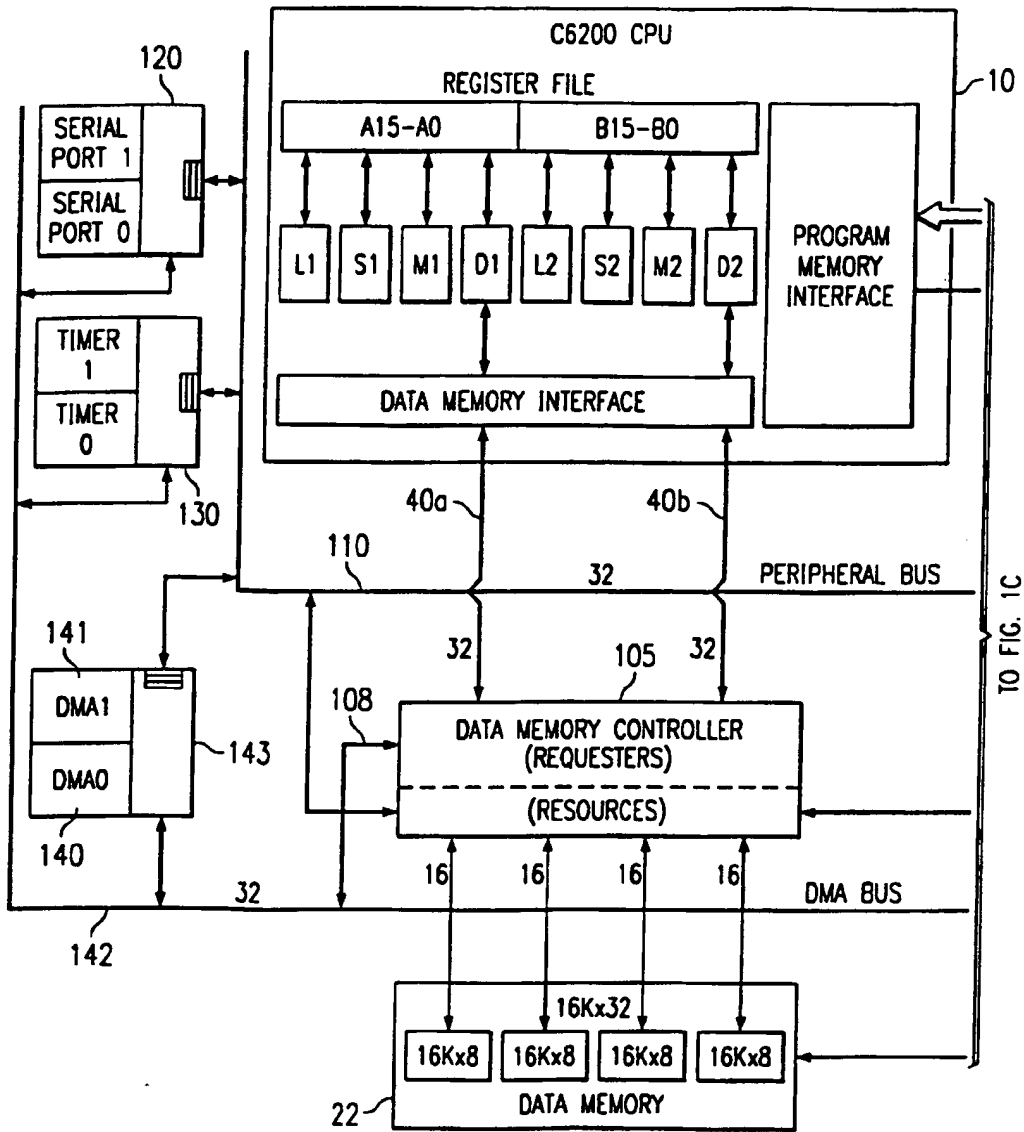
FIG. 1B

FIG. 1C
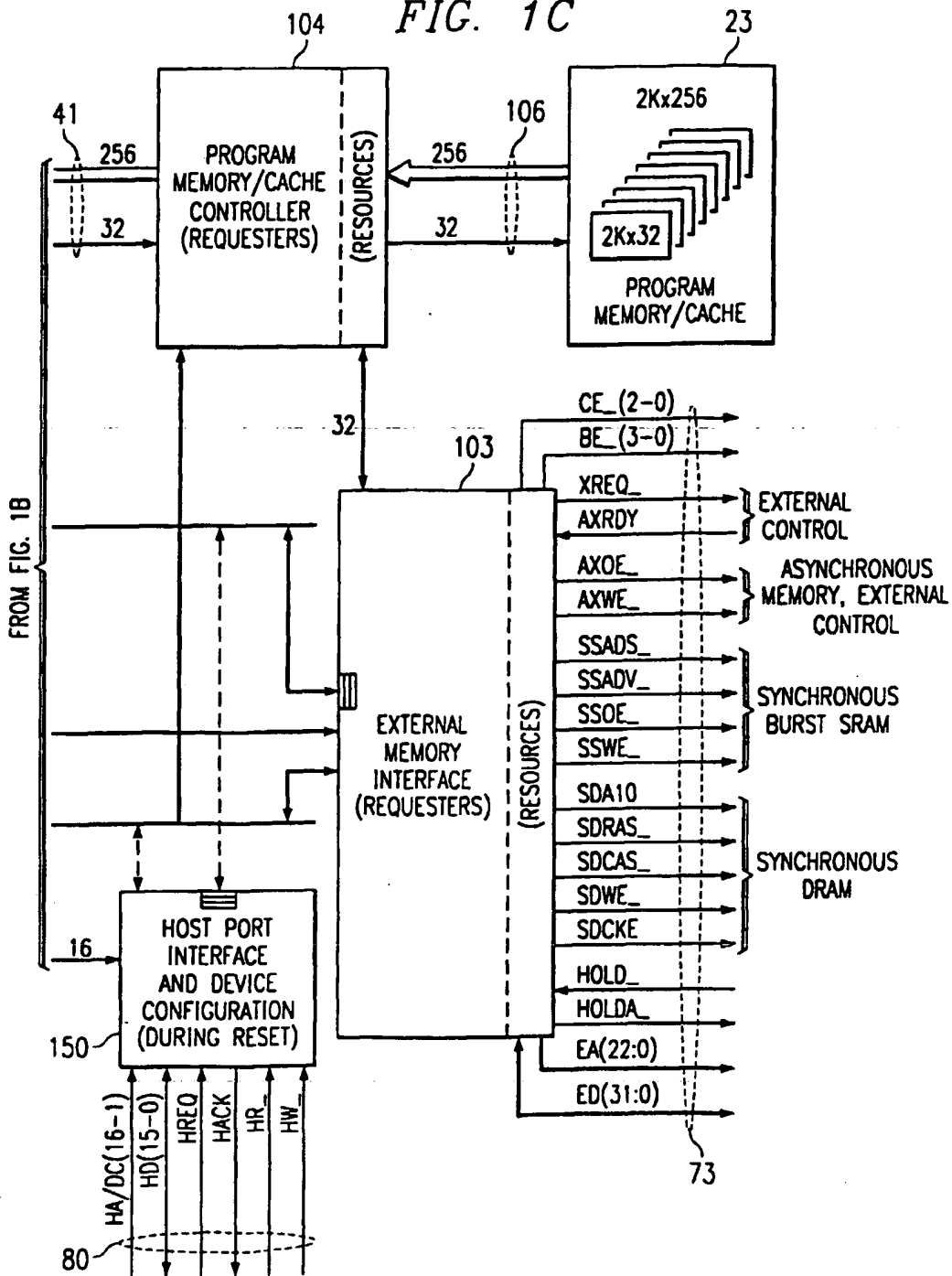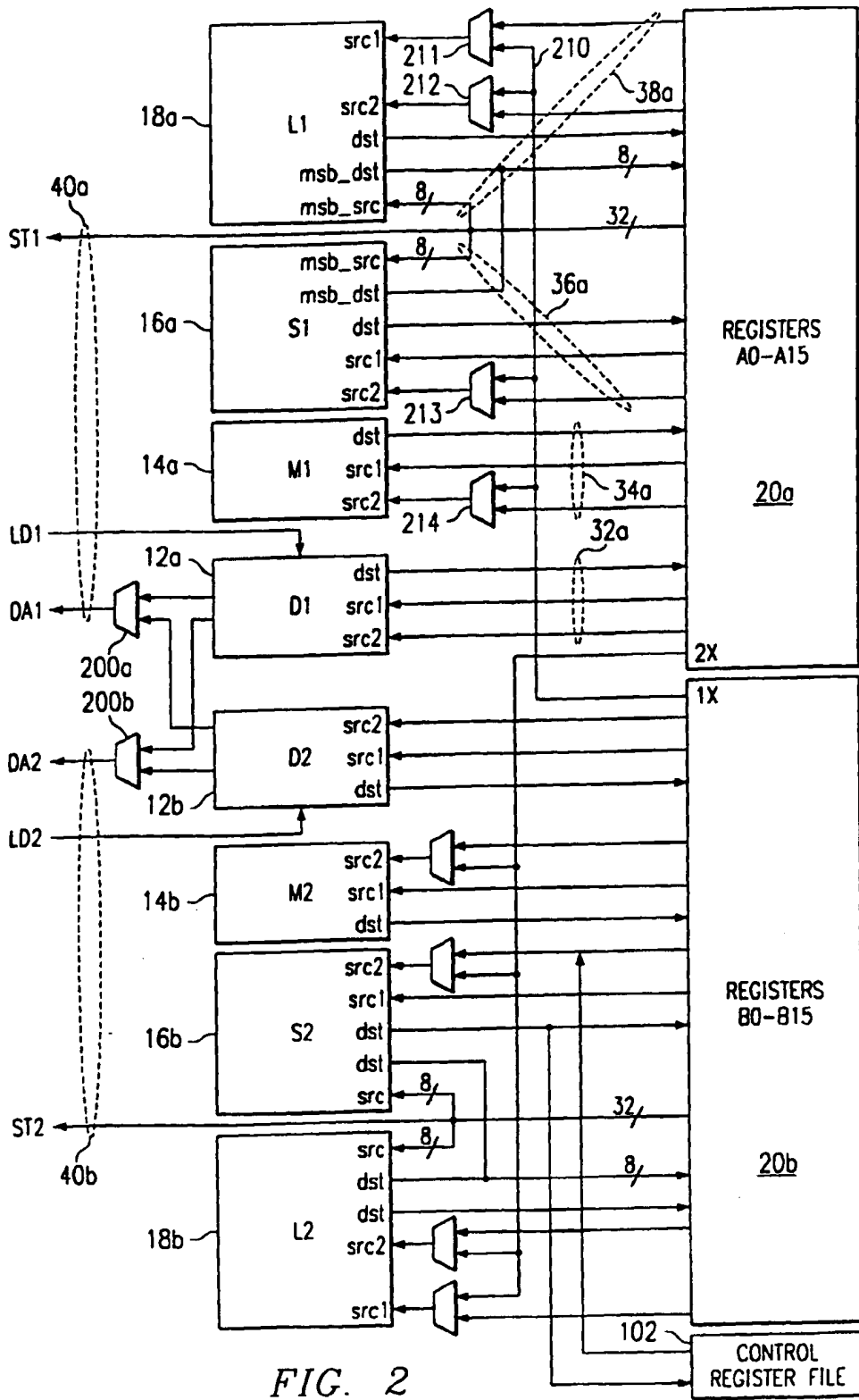
FIG. 2

MEMORY MAP 0

STARTING ADDRESS		BLOCK SIZE (BYTES)
000 0000	EXTERNAL MEMORY SPACE CE0	16M
100 0000	EXTERNAL MEMORY SPACE CE1	4M
140 0000	INTERNAL PROGRAM RAM	64K
141 0000	RESERVED	4M
180 0000	INTERNAL PERIPHERAL SPACE	4M
1C0 0000	RESERVED	4M
200 0000	EXTERNAL MEMORY SPACE CE2	32M
400 0000	RESERVED	1984M
8000 0000	INTERNAL DATA RAM	64K
8001 0000	RESERVED	4M
8040 0000	RESERVED	2044M
1 0000 0000		

FIG. 3A

MEMORY MAP 1

STARTING ADDRESS		BLOCK SIZE (BYTES)
000 0000	INTERNAL PROGRAM RAM	64K
001 0000	RESERVED	4M
040 0000	EXTERNAL MEMORY SPACE CE0	16M
140 0000	EXTERNAL MEMORY SPACE CE1	4M
180 0000	SAME AS MEMORY MAP 0	
1 0000 0000		

FIG. 3B

FIG. 4

DATA MEMORY 22

DATA MEMORY CONTROLLER 105

LENDIAN

CPU 10

PROGRAM MEMORY CONTROLLER

PROGRAM MEMORY/CACHE 23

PERIPHERAL BUS CONTROLLER 111

DMA 143

EMIF 103

TIMERS
INTERRUPT SELECTOR
SERIAL PORTS
HPI CONTROL
DMA CONTROL
EMIF CONTROL

110

130

120

HOST PORT 150

PLL

POWER DOWN

DMA AUXILIARY CHANNEL 160

DMAC0-3

1
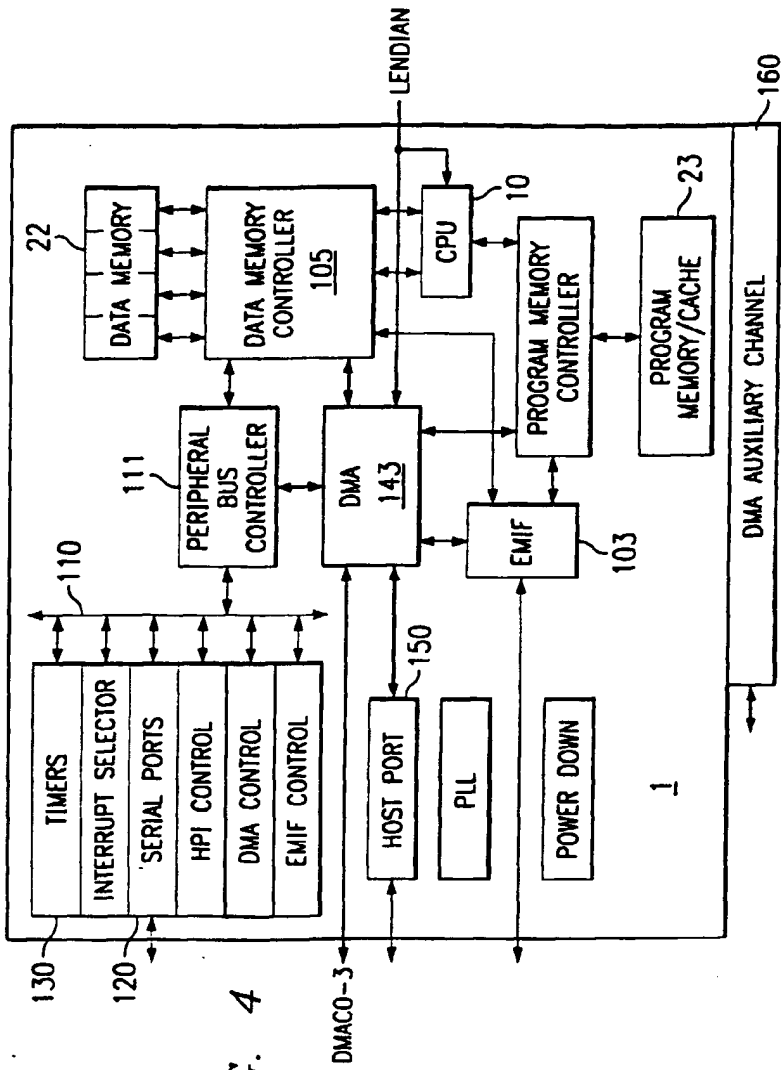
FIG. 5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTSIDE EXTERNAL RANGE. ASSUMED 0.							TAG									BLOCK OFFSET												FETCH PACKET ALIGNMENT. ASSUMED 0.			

FIG. 6

DMA GLOBAL DATA REGISTER

RW, +x

FIG. 7

FIG. 8

31 30	29 28	27	26	25	24	23 ... 19 18 ... 16
DST RELOAD	SRC RELOAD	EMOD	FS	TCINT	PRI	WSYNC RSYNC
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

15 14	13	12	11 10	9 8	7 6	5 4	3 2	1 0
RSYNC	INDEX	CNT RELOAD	SPLIT	ESIZE	DST DIR	SRC DIR	STATUS	START
RW, +0	R, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	R, +0	RW, +0

FIG. 9

31 ... 18	18 17 16
reserved	DMAC EN
R, +0	RW, +0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WSYNC CLR	WSYNC STAT	RSYNC CLR	RSYNC STAT	DROP IE	WDROP COND	RDROP IE	RDROP COND	BLOCK IE	BLOCK COND	LAST IE	LAST COND	FRAME IE	FRAME COND	SX IE	SX COND
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +1	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

FIG. 10

31 ... 16	15 ... 0
ELEMENT COUNT	FRAME COUNT
RW, +0	RW, +0

FIG. 11

31 ... 16	15 ... 0
ELEMENT COUNT RELOAD	FRAME COUNT RELOAD
RW, +0	RW, +0

SOURCE ADDRESS

RW, +x

FIG. 12

DESTINATION ADDRESS

RW, +x

FIG. 13

FRAME INDEX	ELEMENT INDEX
RW, +0	RW, +0

16 15

0

FIG. 14

SPLIT ADDRESS	reserved
RW, +0	R, +0

3 2 0

FIG. 15

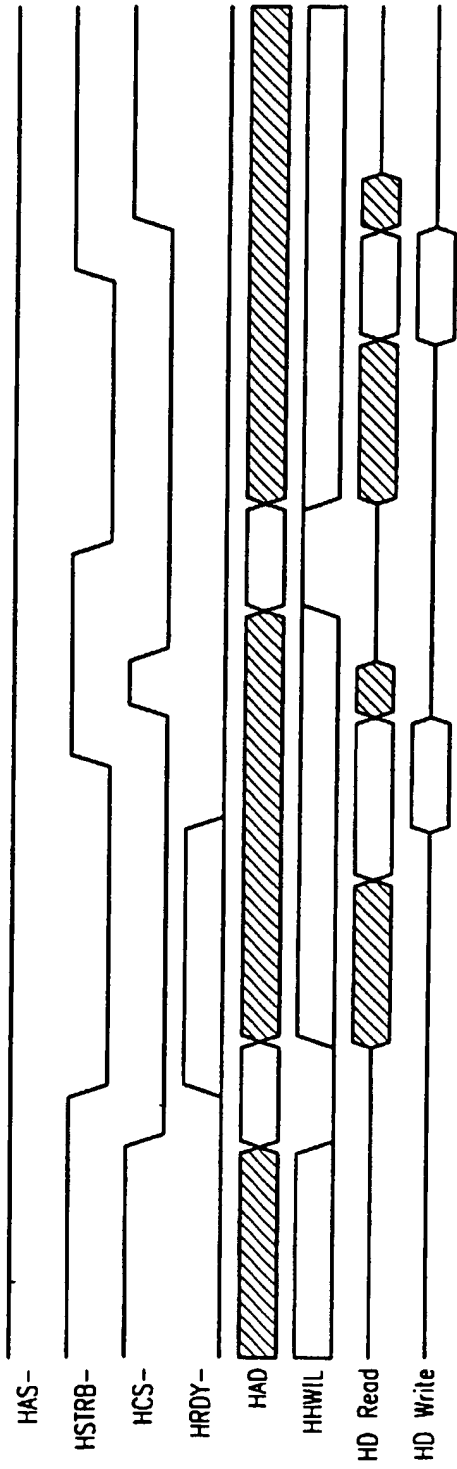| reserved	AUXPRI	CH PRI
R, +0	RW, +0	RW, +0

5 4 3 0

FIG. 16

FIG. 17

FIG. 18

FIG. 20

FIG. 19

FIG. 19A

FIG. 21

FIG. 22

FIG. 23

DMA INTERFACE

DMC INTERFACE

PMC INTERFACE

EXTERNAL
MEMORY
INTERFACE
(EMIF)

→ CLKOUT1
→ CLKOUT2

← ED[31:0]
→ EA[21:2]
→ /CE[3:0]
→ /BE[3:0]

SHARED BY ALL
EXTERNAL
INTERFACES

← /ARDY
→ /AOE
→ /AWE
→ /ARE

ASYNCHRONOUS
SRAM
INTERFACE

→ /SSADS
→ /SSOE
→ /SSWE
→ SSCLK

SBSRAM
INTERFACE

→ /SDRAS
→ /SDCAS
→ /SDWE
→ /SDA10
→ SDCLK

SDRAM
INTERFACE

← /HOLD
→ /HOLDA

BUS HOLD
INTERFACE

INTERNAL PERIPHERAL BUS INTERFACE

EXTERNAL
MEMORY
INTERFACE
(EMIF)

/CEn
CLKOUT2 OR SDCLK
/SDRAS
/SDCAS
/SDWE

/BE[3:0]
EA[13]
SDA10
EA[11:2]
ED[31:0]

VCC

/CS
CLK
/RAS
/CAS
/WE
CKE
DQM[3:0]
A[11]
A[10]
A[9:0]
D[31:0]

16 MBIT
SDRAM

FIG. 28

FIG. 24

31	14	13	12	11	10	9	8	7	6	5	4	3	2	1
reserved	SDCINV	CLK2INV	rsv	/ARDY	/HOLD	/HOLDA	NOHOLD	SDCEN	SSCEN	CLK1EN	CLK2EN	SSCRT	RBTR8	
R, +0	RW, +1	RW, +1	R, +0	R, +x	R, +x	R, +0	RW, +0	RW, +1	RW, +1	RW, +1	RW, +1	RW, +0	RW, +0	

FIG. 25

31	28	27	22	21	20	19	16
WRITE SETUP		WRITE STROBE		WRITE HOLD		READ SETUP	
RW, +1111		RW, +111111		RW, +11		RW, +1111	

FIG. 26

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved				SDWTH	RFEN	INIT		TRCD			TRP				
R, +0				R, +0	R, +1	RW, +1		RW, +0100			RW, +1000				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRC				reserved											
RW, +1111				R, +0											

FIG. 27

31	24	23	12	11	0
reserved		COUNTER			
R, +0		R, +00001000000			

31	12	11	0
PERIOD			
RW, +00001000000			

FIG. 29

FIG. 30

FIG. 31

SDCLK

/CE

SDA10

/SDRAS

/SDCAS

/SDWE

/BE[3:0]

ED[31:0]

COMMAND

DCAB

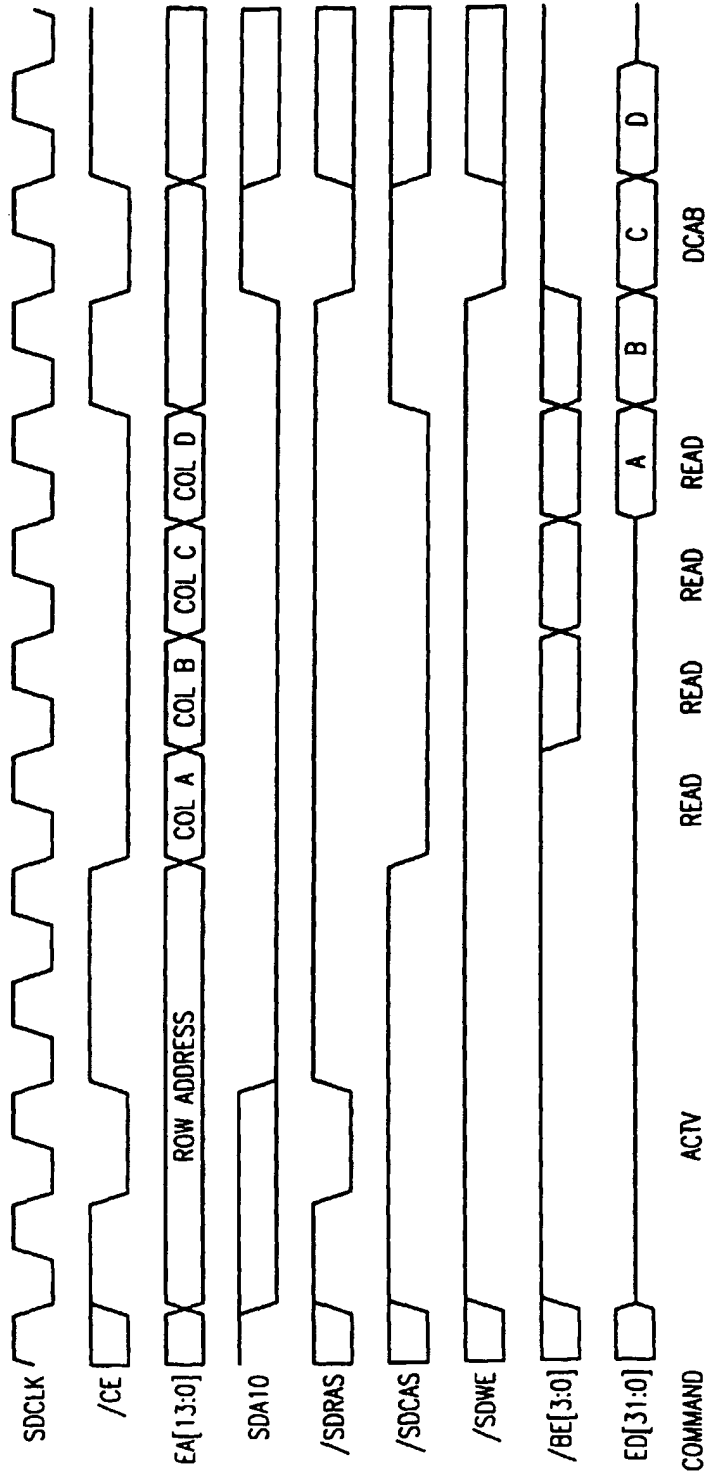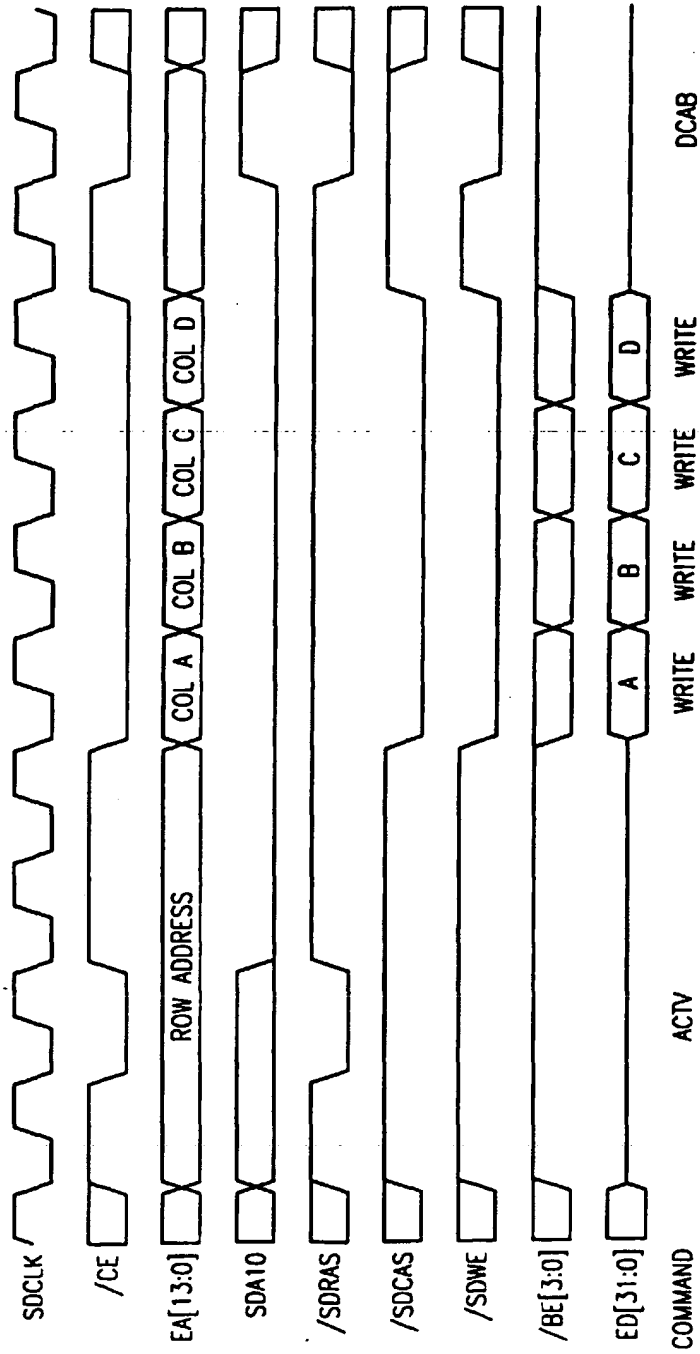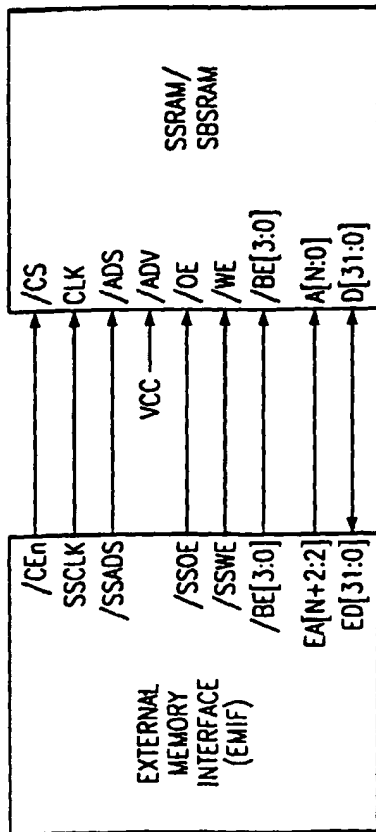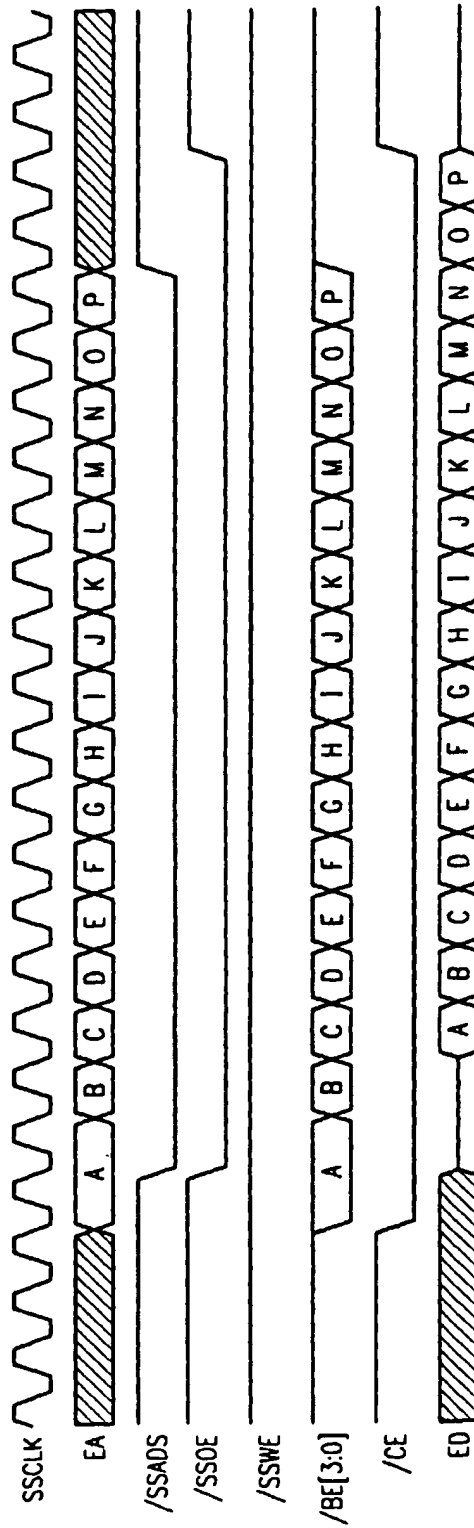*FIG. 32*

FIG. 33

FIG. 34

FIG. 35

FIG. 36

FIG. 37

FIG. 38

EXTERNAL MEMORY INTERFACE (EMIF) — Rx FIFO / Tx FIFO

EXTERNAL MEMORY INTERFACE (EMIF)

/CEn

/AOE

EA[21:2] ──X

ED[31:0]

/ARE

/AWE

Rx FIFO

/RDEN/ $\overline{EN}$

/OE

D[31:0]

RDCLK/RDSTRB

Tx FIFO

/WREN/ $\overline{EN}$

D[31:0]

WRCLK/WRSTRB

FIG. 39

EXTERNAL MEMORY INTERFACE (EMIF)

/CE1

/AOE

EA[N+2:2]

ED[7:0]

/ARE ──X

/ARDY ──┴ GND

ROM

/CS

/OE

A[N:0]

D[7:0]

FIG. 40

EXTERNAL MEMORY INTERFACE (EMIF)

/CE1

/AOE

EA[N+2:2]

ED[15:0]

/ARE ──X

/ARDY ──┴ GND

ROM

/CS

/OE

A[N:0]

D[15:0]

FIG. 41

EXTERNAL MEMORY INTERFACE (EMIF)

/CE1

/AOE

EA[N+2:2]

ED[31:0]

/ARE ──X

/ARDY ──┴ GND

ROM

/CS

/OE

A[N:0]

D[31:0]

FIG. 42

FIG. 43

FIG. 44

FIG. 45

31		24	23	22	21	20	19	18	17	16
reserved			reserved	reserved	XINTM	XSYNCERR		XEMPTY-	XRDY	XRST-
R, +0			R, +0	R, +0	RW, +0	RW, +0		R, +0	R, +0	RW, +0

15	14	13	12	10	9	8	7	6	5	4	3	2	1	0
DLB		RJUST		CLKSTP		reserved		reserved		RINTM		RSYNCERR	RRDY	RRST-
RW, +0		RW, +0		RW, +0		R, +0		R, +0		RW, +0		RW, +0 RFULL R, +0	R, +0	RW, +0

FIG. 46

31		16
reserved		
R, +0		

15 14 13	12	11		10	9		8	7 6	5	4	3	2	1	0
resv	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM	resv	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP
R, +0	RW, +0	RW, +0 RW, +0	RW, +0	RW, +0 RW, +0	R, +0	RW, +0	RW, +0	R, +0	RW, +0	RW, +0 RW, +0	RW, +0

FIG. 47

31	30	24 23	21 20	19 18	17	16
RPHASE		RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY
RW, +0		RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

15		14	8 7	5 4	0
reserved		RFRLEN1		RWDLEN1	reserved
R, +0		RW, +0		RW, +0	R, +0

130

FIG. 48

31	30		24	23			20	19		18	17			16
XPHASE	XFRLEN2			XWDLEN2				XCOMPAND		XFIG	XDATDLY			
RW, +0	RW, +0			RW, +0				RW, +0		RW, +0	RW, +0			

15	14		8	7			5	4				0
reserved	XFRLEN1			XWDLEN1				reserved				
R, +0	RW, +0			RW, +0				R, +0				

FIG. 49

CLK(R/X)_int

FS(R/X)_int

D(R/X): A1 A0 ... B7 B6 B5 B4 B3 B2 B1 B0

FIG. 50

CLKR_int

DR: B7 B6

DATA SETUP — DATA HOLD
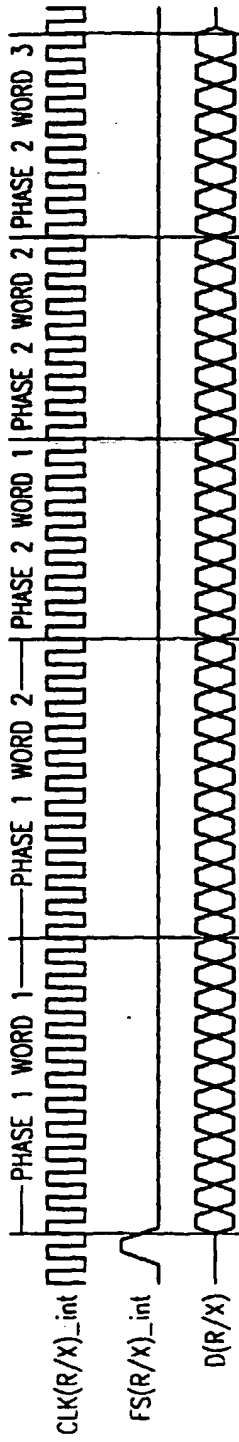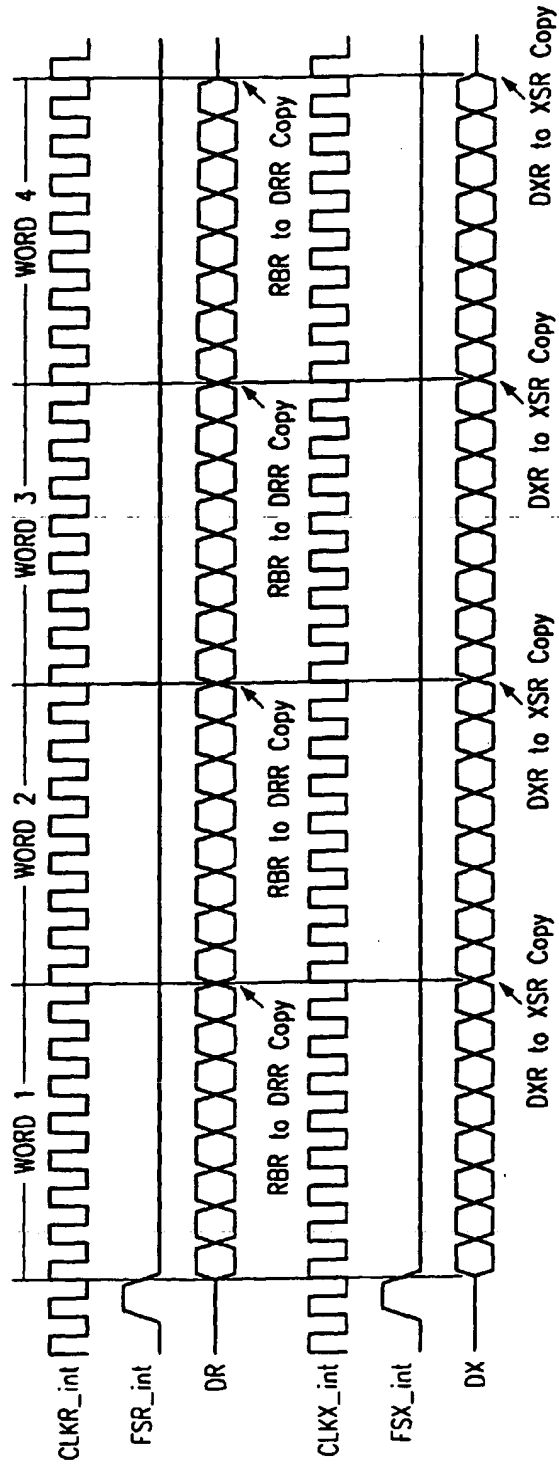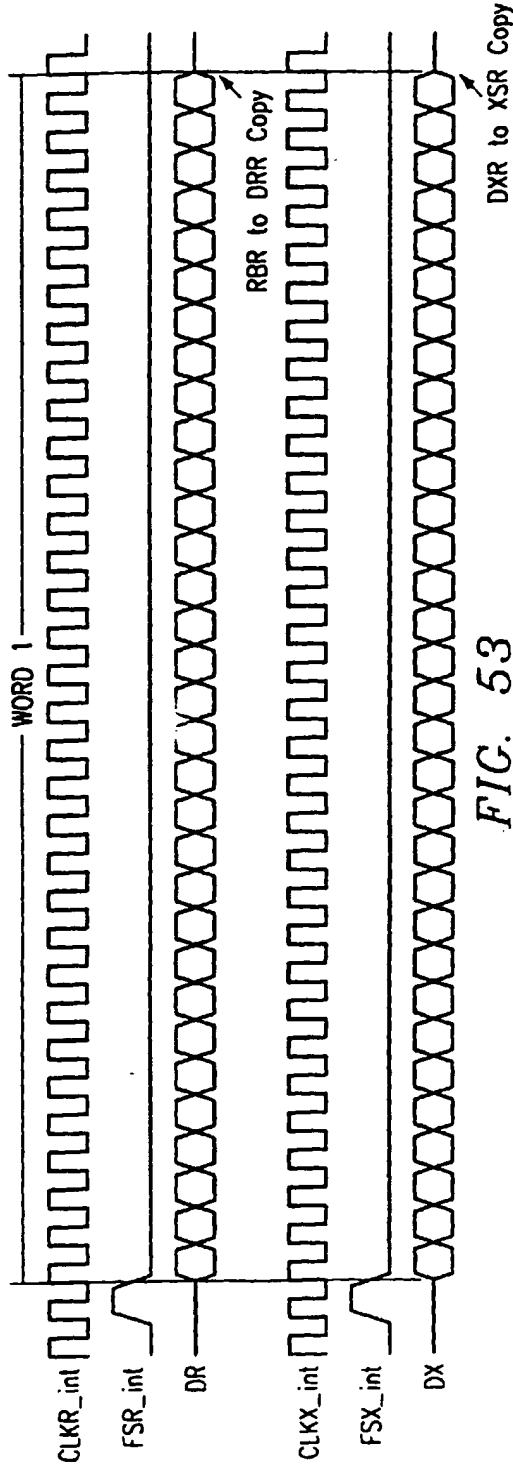
FIG. 51

FIG. 52

FIG. 53

FIG. 54

FIG. 55

FIG. 56

$PxWy = Phase \; x \; Word \; y$

FIG. 57

$PxWyBz = Phase \; x \; Word \; y \; Bit \; z$

FIG. 58

FIG. 59

CLKX_int

FSX_int

DX [A1] [A0] ─── [B7] [B6] [B5] [B4] [B3] [B2] [B1] [B0] ─── [C7] [C6] [C5]

XRDY

Write of DXR   DXR-to-XSR copy   Write of DXR

FIG. 60

CLK(R/X)_int

FS(R/X)_int

D(R/X) [A2] [A1] [A0] [B7] [B6] [B5] [B4] [B3] [B2] [B1] [B0] [C7] [C6]

FIG. 61

FIG. 62

FIG. 63

FIG. 64

CLK(R/X)_int

FS(R/X)_int

D(R/X)   A0 B7 B6 B5 B4 B3 B2 B1 B0 C7 C6 C5 C4

FRAME SYNCHRONIZATION IGNORED

(R/X)SYNCERR

FIG. 65

CLKR_int

FSR_int

DR   A1 A0 B7 B6 B5 B4 B3 B2 B1 B0 C7 C6 C5

RRDY

RBR-to-DRR copy (A)   No RBR-to-DRR copy (B)
No Read of DRR (A)   Read of DRR (A)

RFULL

FIG. 66

FIG. 67

CLKR_int

FSR_int

DR A1 A0 B7 B6 B5 B4 B3 B2 B1 B0 C7 C6 C5

RRDY

RBR-to-DRR copy (A)   No RBR-to-DRR copy (B)   RBR-to-DRR copy (B)

No Read of DRR (A)   Read of DRR (A)

RFULL

FIG. 68

RECEIVE FRAME SYNC PULSE OCCURS

IS RFULL SET?

NO

YES

CASE 1: RECEIVE OVERRUN
IGNORE FRAME PULSE.
RECEIVER STAYS HALTED.

UNEXPECTED FRAME SYNC PULSE?

NO

YES

RFIG=1?

NO

YES

CASE 2: WITH FRAME IGNORE
IGNORE FRAME PULSE.
RECEIVER CONTINUES RUNNING.

CASE 3: NORMAL RECEPTION
START RECEIVING DATA.

CASE 4: WITHOUT FRAME IGNORE
ABORT RECEPTION. SET RSYNCERR.
START NEXT RECEPTION AFTER
CLEARING RSYNCERR. NO RBR TO DRR
TRANSFER. THUS 1 WORD IS LOST.

FIG. 69

FIG. 70

CLKX_int

FSX_int

DX [A1 X A0]——[B7 X B6 X B5 X B4 X B3 X B2 X B1 X B0]——[B7 X B6 X B5 X B4]

XRDY   DXR-to-XSR copy   Write of DXR

XEMPTY-

FIG. 71

CLKX_int

FSX_int

DX [A1 X A0]——[B7 X B6 X B5 X B4 X B3 X B2 X B1 X B0]——[C7 X C6 X C5]

XRDY   DXR-to-XSR copy   Write of DXR (C)   DXR-to-XSR copy (C)

XEMPTY-

FIG. 72

FIG. 73

TRANSMIT FRAME SYNC PULSE OCCURS

UNEXPECTED FRAME SYNC PULSE?

— NO → CASE 2: NORMAL TRANSMISSION START NEW TRANSMIT

— YES →

XFIG=1?

— NO → CASE 3: WITHOUT FRAME IGNORE ABORT TRANSFER. SET XSYNCERR. RESTART CURRENT TRANSFER.

— YES → CASE 1: WITH FRAME IGNORE IGNORE FRAME PULSE. TRANSMITTER STAYS RUNNING.

FIG. 74

CLKX_int

FSX_int

DX   A1 A0   B7 B6 B5 B4   B7 B6 B5 B4 B3 B2 B1 B0

XRDY

DXR-to-XSR copy (B)   Write of DXR (C)   DXR-to-XSR (C)   Write of DXR (D)

XSYNCERR

UNEXPECTED FRAME SYNCHRONIZATION

DR → [RSR] → [RBR] → 8/ → [EXPAND] → 16/ → [RJUST] → 32/ → [DRR] → TO CPU/DMA

DX ← [XSR] ← 8/ ← [COMPRESS] ← 16/ ← [DXR] ← FROM CPU/DMA

FIG. 75

μ−LAW   15   2   1   0   A−LAW   15   3   2   0

| VALUE | 0 | | VALUE | 0 |

FIG. 76

DR → [RSR] → [RBR] → [EXPAND] → [RJUST] → [DRR] → TO CPU/DMA

2 (DLB)   1

DX ← [XSR] ← [COMPRESS] ← [DXR] ← FROM CPU/DMA

FIG. 77

CPU CLOCK → 1

CLKSP → [⊕] → 0 → [÷] CLKGDV → [÷] FPER → [FRAME PULSE] FWID → FSG

CLKSM

→ CLKG

CLKS → [FRAME PULSE DETECTION AND CLOCK SYNCHRONIZATION]

GSYNC →

FSR →

FIG. 78

144

FIG. 79

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GSYNC	CLKSP	CLKSM	FSGM								FPER				
RW, +0	RW, +0	RW, +0	RW, +0								RW, +0				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				FWID							CLKGDV				
				RW, +0							RW, +0				

FIG. 80

CLKS (CLKSP=0)

CLKS (CLKSP=1)

FSR external (FSRP=0)

FSR external (FSRP=1)

CLKG(no need to re-sync)

CLKG(needs to re-sync)

FSG

FIG. 81

CLKS (CLKSP=0)

CLKS (CLKSP=1)

FSR external (FSRP=0)

FSR external (FSRP=1)

CLKG(no need to re-sync)

CLKG(needs to re-sync)

FSG

FIG. 82

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

CLKG

FRAME PERIOD: (FPER+1) x CLKG

FRAME WIDTH: (FWID+1) x CLKG

FSG

FIG. 83

FIG. 84

FIG. 85

4.096MHz CLKS

FSR external

FSG, FSR_int, FSX_int

2.048MHz CLKG, CLKR_int, CLKX_int (first FSR)

DR, DX (first FSR)

CLKG, CLKR_int, CLKX_int (subsequent FSR)

DR, DX (subsequent FSR)

W1B7 W1B6 W1B5 W1B4 W1B3 W1B2 W1B1 W1B0 W2B7

W32B0 W1B7 W1B6 W1B5 W1B4 W1B3 W1B2 W1B1 W1B0 W2B7

WxBy=Word x Bit y

CLKS

FSR external

FSG, FSR_int, FSX_int

CLKG, CLKR_int, CLKX_int (first FSR)

DR, DX (first FSR)

CLKG, CLKR_int, CLKX_int (subsequent FSR)

DR, DX (subsequent FSR)

W1B7 W1B6 W1B5 W1B4 W1B3 W1B2 W1B1 W1B0 W2B7

W32B0 W1B7 W1B6 W1B5 W1B4 W1B3 W1B2 W1B1 W1B0 W2B7

WxBy=Word x Bit y

FIG. 86

CLKS

FS(R/X)_int

CLK(R/X)_int

D(R/X) | W32B0 | W1B7 | W1B6 | W1B5 | W1B4 | W1B3 | W1B2 | W1B1 | W1B0 | W2B7 |

WxBy=Word x Bit y

FIG. 87

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		reserved					XPBBLK		XPABLK		XCBLK		XMCM	
		R, +0					RW, +0		RW, +0		R, +0		RW, +0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		reserved					RPBBLK		RPABLK		RCBLK		reserved	RMCM
		R, +0					RW, +0		RW, +0		R, +0		R, +0	

FIG. 88

PARTITION A | 0-15 | 32-47 | 64-79 | 96-111 |

PARTITION B | 16-31 | 48-63 | 80-95 | 112-127 | 0-15 |

FS(R/X)

FIG. 89A

XMCM=00b

FSX_int

DX

XRDY

W0, W1, W2, W3

Write of DXR (W1)

Write of DXR (W2)

Write of DXR (W3)

DXR-to-XSR (W0)

DXR-to-XSR copy (W1)

DXR-to-XSR copy (W2)

DXR-to-XSR copy (W3)

FIG. 89B

XMCM=01b; XPABLK=00b, XCER=1010b

FSX_int

DX

XRDY

W1, W3

Write of DXR (W3)

DXR-to-XSR copy (W1)

DXR-to-XSR copy (W3)

FIG. 89C

XMCM=10b; XPABLK=00b, XCER=1010b

FSX_int

DX

XRDY

W1, W2, W3

Write of DXR (W1)

Write of DXR (W2)

Write of DXR (W3)

DXR-to-XSR (W0)

DXR-to-XSR copy (W1)

DXR-to-XSR copy (W2)

DXR-to-XSR copy (W3)

EP 0 901 081 A2

XMCM=11b; RPABLK=00b, XPABLK=X, RCER=1010b, XCER=1000b

FS(R/X)_int

DR

W1   W3

RRDY

Read of DRR (W3)

RBR-to-DRR copy (W3)

Read of DRR (W1)

RBR-to-DRR copy (W1)

RBR-to-DRR copy (W3)

DX

W3

XRDY

DXR-to-XSR copy (W1)

Write of DXR (W3)

DXR-to-XSR copy (W3)

FIG. 89D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCEB15	RCEB14	RCEB13	RCEB12	RCEB11	RCEB10	RCEB9	RCEB8	RCEB7	RCEB6	RCEB5	RCEB4	RCEB3	RCEB2	RCEB1	RCEB0
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCEA15	RCEA14	RCEA13	RCEA12	RCEA11	RCEA10	RCEA9	RCEA8	RCEA7	RCEA6	RCEA5	RCEA4	RCEA3	RCEA2	RCEA1	RCEA0
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

FIG. 90

151</sub>

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XCEB15	XCEB14	XCEB13	XCEB12	XCEB11	XCEB10	XCEB9	XCEB8	XCEB7	XCEB6	XCEB5	XCEB4	XCEB3	XCEB2	XCEB1	XCEB0
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCEA15	XCEA14	XCEA13	XCEA12	XCEA11	XCEA10	XCEA9	XCEA8	XCEA7	XCEA6	XCEA5	XCEA4	XCEA3	XCEA2	XCEA1	XCEA0
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

FIG. 91

CLK(R/X)_int CLKSTP='b100

CLK(R/X)_int CLKSTP='b101

CLK(R/X)_int CLKSTP='b110

CLK(R/X)_int CLKSTP='b111

D(R/X) B7 B6 B5 B4 B3 B2 B1 B0

FS(R/X)_int

FIG. 92

PERIPHERAL BUS TO
CPU AND DMA

COUNT
ZERO

GO

HLD—

COUNT
ENABLE

TIMER
COUNTER
REGISTER

TIMER
PERIOD
REGISTER

EDGE DETECT

CLKSRC

EQUALS COMPARATOR

1   0

CPU
CLOCK
/4

PULSE GENERATOR

C/P—
DATOUT

TSTAT, TIMER OUTPUT
TINT, TIMER INTERRUPT
TO CPU AND DMA

0   1

FUNC

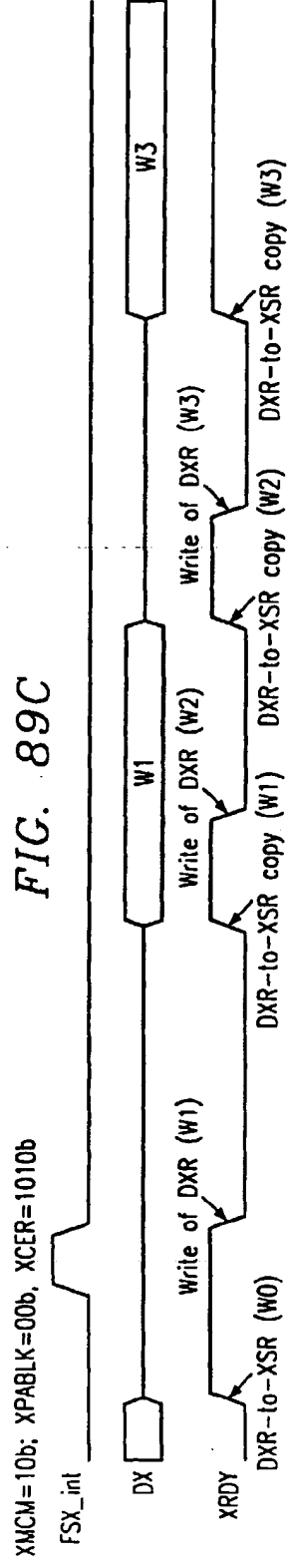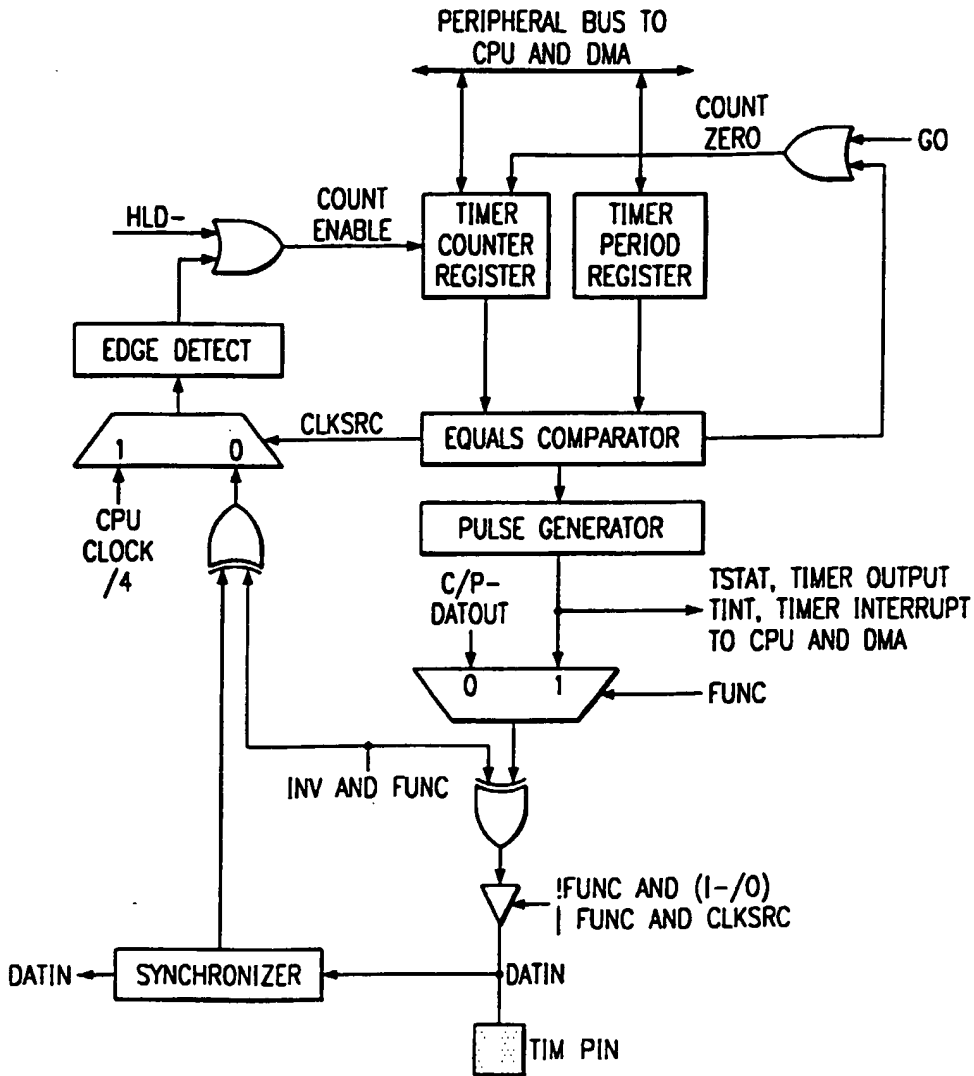INV AND FUNC

!FUNC AND (I—/0)
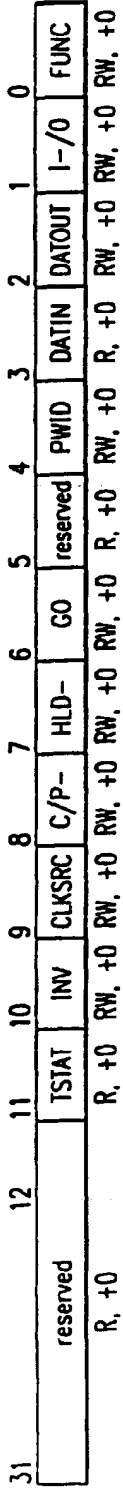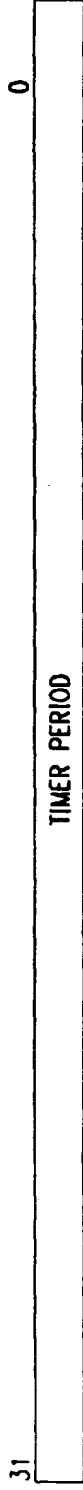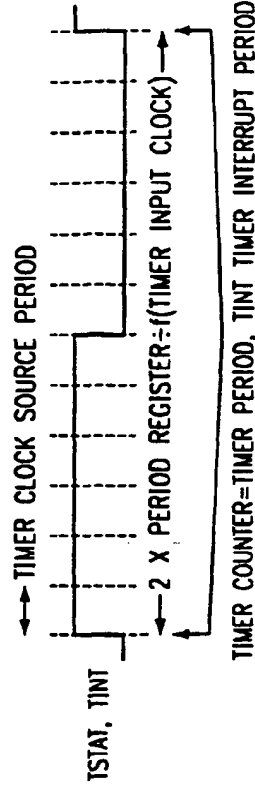| FUNC AND CLKSRC
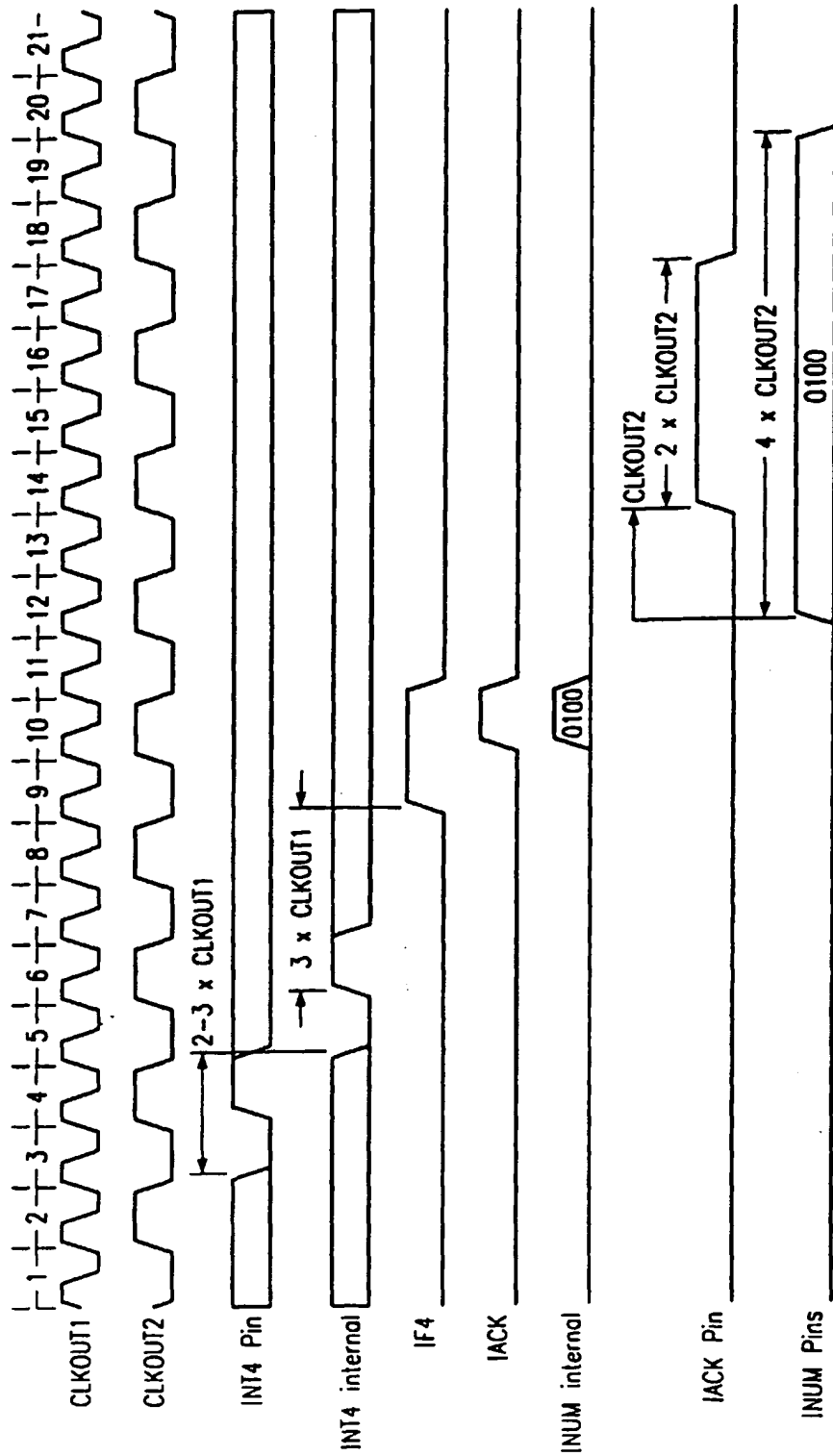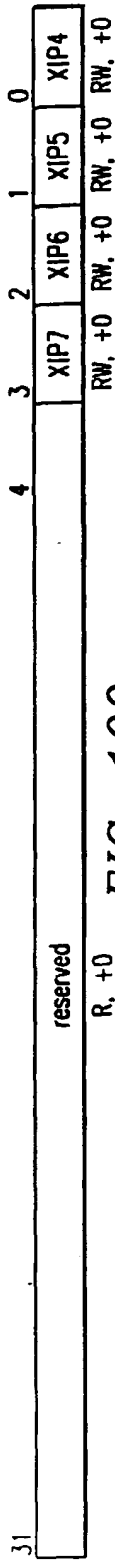
DATIN

SYNCHRONIZER

DATIN

TIM PIN

FIG. 93

FIG. 94

FIG. 95

FIG. 96

FIG. 97

FIG. 98

CLKOUT1

CLKOUT2

INT4 Pin

INT4 internal

IF4

IACK

INUM internal

IACK Pin

INUM Pins

0100

2-3 x CLKOUT1

3 x CLKOUT1

CLKOUT2

2 x CLKOUT2

4 x CLKOUT2

0100

FIG. 99

FIG. 100

31			4	3	2	1	0
reserved				XIP7	XIP6	XIP5	XIP4
R, +0				RW, +0	RW, +0	RW, +0	RW, +0

FIG. 101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved		INTSEL9				reserved		INTSEL8			reserved		INTSEL7		
R, +0		RW, +1001				R, +0		RW, +1000			R, +0		RW, +0111		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved		INTSEL6				reserved		INTSEL5			reserved		INTSEL4		
R, +0		RW, +0110				R, +0		RW, +0101			R, +0		RW, +0100		

FIG. 102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved		INTSEL15				reserved		INTSEL14			reserved		INTSEL13		
R, +0		RW, +0010				R, +0		RW, +0001			R, +0		RW, +0000		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved		INTSEL12				reserved		INTSEL11			reserved		INTSEL10		
R, +0		RW, +1011				R, +0		RW, +1010			R, +0		RW, +0011		

FIG. 105

31	16	15	14	13	12	11	10	9	0
• • •		reserved	ENABLED OR NON-ENABLED INTERRUPT WAKE[1]	ENABLED INTERRUPT WAKE[1]	Pd3	Pd2	Pd1	• • •	CSR

PRWD FIELD OF CSR REGISTER

156

CLKOUT1 FREQUENCY RANGE 240-300 MHz- 1 0 0
CLKOUT1 FREQUENCY RANGE 180-240 MHz- 0 1 1
CLKOUT1 FREQUENCY RANGE 120-180 MHz- 0 1 0
CLKOUT1 FREQUENCY RANGE 60-120 MHz- 0 0 1
CLKOUT1 FREQUENCY RANGE 0-60 MHz- 0 0 0

3.3V

20K Ohms

2.5V

EMI FILTER

.01μF (BYPASS)

10μF (BYPASS)

R1

C1

C2

PLLV

PLLF

PLLG

CLKIN

PLLFREQ3

PLLFREQ2

PLLFREQ1

RSV2

CLKOUT1

CLKMODE0

CLKMODE1

1 1 -MULTx4 f(CLKOUT)=f(CLKIN)x4
0 1 -RESERVED
1 0 -RESERVED
0 0 -MULTx1 f(CLKOUT)=f(CLKIN)

FIG. 103

CLKOUT1

INTERNAL CLOCK TIME

Pd1

Pd3

PD_

CLOCK PLL

POWER-DOWN LOGIC

IFR

IER

PWRD CSR

CPU

INTERNAL PERIPHERAL

INTERNAL PERIPHERAL

Pd2

CLKIN

RESET_

FIG. 104

(72) Inventors:
• Nguyen, Tai H.
Houston, Texas 77082 (US)

• Jones, Jason A.T.
Houston, Texas 77042 (US)
• Bradley, Jonathan G.
Houston, Texas 77083 (US)
• Seshan, Natarajan
Houston Texas 77063-1234 (US)
• Quay, Jeffrey R.
Royse City Texas 75189 (US)
• Williams, Kenneth L.
Sherman, Texas 75090 (US)
• Moody, Michael J.
McKinney, Texas 75070 (US)
• Simar, Laurence R.Jr.
Richmond, Texas 77469 (US)
• Scales, Richard H.
06271 Villeneuve-Loubet (FR)

(74) Representative: Potter, Julian Mark et al
D. Young & Co.,
21 New Fetter Lane
London EC4A 1DA (GB)

(54) A digital signal processor with peripheral devices and external interfaces

(57) A Digital Signal Processor is described which has a variety of on-chip peripheral devices and a variety of external interfaces. The peripheral devices include a multi-channel serial port, a multi-channel direct memory interface, and a timer. The external interfaces include a host port interface and an extended memory interface.

FIG. 1A

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	GB 2 200 818 A (INTEL CORP) 10 August 1988 (1988-08-10) * page 1, paragraph 1 - paragraph 2 *	1,2	G06F13/42 G06F9/38
Y	* page 8, paragraph 3 - page 19, paragraph 3 * ---	3-7	
Y	WILSON: "Communications Controllers Search for Higher Integration" COMPUTER DESIGN., vol. 28, no. 19, 1 October 1989 (1989-10-01), page 28,30,32 XP000065774 PENNWELL PUBL. LITTLETON, MASSACHUSETTS., US ISSN: 0010-4566 * page 28, left-hand column, paragraph 4 - right-hand column, paragraph 3 * ---	3-7	
A	EP 0 266 790 A (NEC CORP) 11 May 1988 (1988-05-11) * page 4, column 5, line 43 - page 5, column 7, line 43; figures 1,2,4 * ---	1,2	TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F
A	US 5 590 371 A (TASHIRO) 31 December 1996 (1996-12-31) * column 2, line 35 - column 5, line 5; figures 3,4 * -----	1,2	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	18 February 2000	GILL, S

European Patent
Office

CLAIMS INCURRING FEES

The present European patent application comprised at the time of filing more than ten claims.

☐ Only part of the claims have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims and for those claims for which claims fees have been paid, namely claim(s):

☐ No claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims.

LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

see sheet B

☐ All further search fees have been paid within the fixed time limit. The present European search report has been drawn up for all claims.

☐ As all searchable claims could be searched without effort justifying an additional fee, the Search Division did not invite payment of any additional fee.

☐ Only part of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the inventions in respect of which search fees have been paid, namely claims:

[X] None of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the invention first mentioned in the claims, namely claims:

1-7

		Application Number
European Patent **Office**	**LACK OF UNITY OF INVENTION** **SHEET B**	EP 98 30 5453

The Search Division considers that the present European patent application does not comply with the requirements of unity of invention and relates to several inventions or groups of inventions, namely:

1. Claims: 1-7

    Processing device with serial port having dual phase frames

2. Claims: 8-19

    Processing device having a memory controller

3. Claims: 20,21,31

    Processing device having a direct memory access controller with DMA interrupt circuitry

4. Claims: 22-26

    Processing device having a direct memory access controller with programmable increment/decrements

5. Claim : 27

    Processing device having selectable frame synchronization events

6. Claims: 28-29

    Processing device having a direct memory access controller with auto-initialization circuitry

7. Claim : 30

    Processing device having direct memory access controller with output status indication circuitry

8. Claims: 32-34

    Processing device having an interrupt selector

**ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.**   EP 98 30 5453

18-02-2000

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
GB 2200818	A	10-08-1988	US	4780814 A	25-10-1988
			JP	2709820 B	04-02-1998
			JP	63296540 A	02-12-1988
EP 266790	A	11-05-1988	JP	63118856 A	23-05-1988
			DE	3751083 D	30-03-1995
			DE	3751083 T	19-10-1995
			US	5025414 A	18-06-1991
US 5590371	A	31-12-1996	JP	7129486 A	19-05-1995
			DE	4437959 A	04-05-1995